SR-1
JULY 1990

(2)

USER'S MANUAL FOR A SEMI-SPECTRAL PRIMITIVE EQUATION REGIONAL OCEAN-CIRCULATION MODEL VERSION 3.0

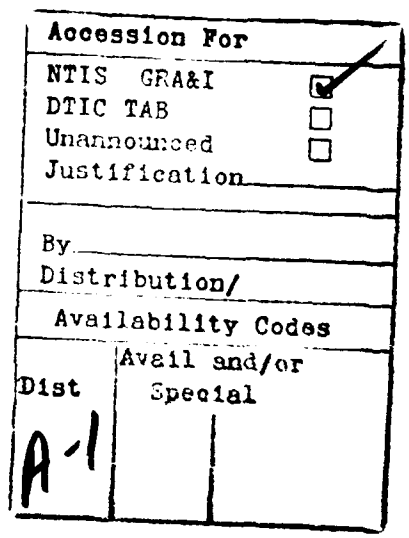
KATHERINE S. HEDSTROM
INSTITUTE FOR NAVAL OCEANOGRAPHY

These working papers were prepared for the timely dissemination of information; this document does not represent the official position of INO.

Acknowledgments

I would like to thank Dale Haidvogel, whose patience in describing the SPEM to me made this document possible, and who made many helpful suggestions while I was writing it. I would also like to thank him for writing up the model physics and for allowing me to include that description here. Thanks go to Hsiao-Ming Hsu for his careful reading of an early version of this user's guide.

Aike Beckmann has been very helpful in my efforts to upgrade this document, both in contributing information about model changes and in carefully checking what I have written.



The Institute for Naval Oceanography is sponsored by NOARL. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of NOARL.

Abstract

The Semi-spectral Primitive Equation Model (SPEM), authored by Dr. Dale Haidvogel of the Johns Hopkins University, is one approach to regional and basin scale ocean modeling. This user's manual for SPEM describes the model equations as well as what the user must do to configure the model for a specific application.

Table of Contents

List of Figures

Chapter 1

Introduction

This user's manual for the semi-spectral primitive equation model (SPEM) attempts to describe the model equations as well as what the user has to do to configure the model for a specific application. Some initial tests of the SPEM are described in Haidvogel, Wilkin and Young[8].

The principle attributes of the model are as follows:

General

- Primitive equations with temperature, salinity and an equation of state.
- Hydrostatic and Boussinesq approximations.
- Second-moment conservation.
- Optional Lagrangian floats.

Horizontal

- Orthogonal-curvilinear coordinates.
- Arakawa C grid.
- Periodic channel or user supplied boundary conditions.

Vertical

- Sigma (bottom-topography following) coordinate.
- Chebyshev modal expansion.
- Rigid lid on top.

Chapters 2 and 3 describe the model physics and numerical techniques and are an update of the description in Haidvogel, Wilkin, and Young[8]. Chapter 4 lists the model subroutines, functions and variables. As distributed, SPEM is ready to run a seamount resonance problem. The process of configuring SPEM for a particular application is described in chapter 5, including a discussion of the seamount application. Section 6 describes the Lagrangian floats and how to use them, including the plotting post-processor **fltplt**.

The SPEM uses version 3.0 of the NCAR graphics package to plot the model fields during execution. You must have version 3.0 if you want to use the SPEM plotting subroutines as they are. You will also have to edit the subroutine **velvct** as described in section 4.5 in order to get the velocity vectors to plot. In the main part of SPEM, the graphics is isolated to calls to **opngks**, **peplt** (twice), **ploth** and **clsgks**, and could be commented out if desired.

1.1 Acquiring the SPEM code

The version of the model described in this document is available over Internet via anonymous ftp (user ftp, any password) from jupiter.ino.ucar.edu (IP number 128.160.2.21). Once you are connected, type 'cd spem3.0' and then 'dir' to get a directory listing. You will find the files:

Readme	Read this first
Makefile	For a UNIX environment
spem.com	For a VMS environment
spem.input	The input text file
spem3.0.whole	The model code without include directives
spem3.0.f	The model code with include directives
spem3.0.h	The SPEM include file
mud2.f	The multigrid solver
mud2.doc	The documentation for **mud2**
saxpy.f	Cray library routines
ezgrid.f	A program for making a simple rectangular grid
fltplt.whole	A program for making color plots from the float history files (without includes)
fltplt.f	A program for making color plots from the float history files (with includes)
fltplt.h	The fltplt include file
grid	A subdirectory which has the grid-generation programs described in Hedstrom and Wilkin[10].

mud2 is copyrighted by NCAR and they request that you not make any changes to it without their permission.

If you are unable to acquire the code in this fashion feel free to contact me at:

Kate Hedstrom
Institute for Naval Oceanography
Building 1103
Stennis Space Center, MS 39529
(601)-688-5737

Internet: kate@ncar.ucar.edu
SPAN: NSFGW::"ncar.ucar.edu!kate"

1.2 Future improvements

A number of improvements to the model are being worked on or need to be tested more thoroughly before being released. These include:

- A first and second moment diagnostics package.

- A plotting post-processor.
- A netCDF interface. It will will be used for porting binary data files from one machine to another and as an interface to the ECMOP facility at INO.
- A surface mixed-layer in SPEM. One option which has been investigated is the Price-Weller-Pinkel mixed layer model.
- A data assimilation scheme in SPEM.
- A program for generating polynomials other than the Chebyshevs. Although alternate polynomials do very well in a flat-bottomed configuration, we are still exploring their behavior in the presence of topography.
- A free sea surface.
- A method for masking out land areas.
- Multitasking on 4-16 Cray processors.

1.3 Changes between version 2.0 and version 3.0

Some quite substantial changes have been made to the model, including:

- By default, the SPEM conserves second moments to within a few percent when the energy-containing scales of motion are well resolved. There is now an option to conserve second moments exactly.
- The old implementation of the horizontal pressure gradient terms only works for moderate bottom slopes (2×10^{-2} maximum). Some corrections can be added to the pressure gradients to more accurately deal with steep topography. However, these corrections are not guaranteed to solve all problems with steep topography.
- Separate equations for temperature and salinity and an equation of state are now used instead of the density equation. If density is a function of temperature only, the calculation of salinity can be turned off.
- There is an option for calculating the two components of the surface pressure gradient.
- The option of vertically mixing regions of static instability has been added.
- The vertical derivatives and integrals using model polynomials have been rewritten and are now much faster on vector computers.
- Subroutines for following Lagrangian floats have been written. There is also a program which makes color plots of the float tracks.
- The timestepping has been changed again. See Appendix A.

- The problem which SPEM is set up to run as distributed has been changed from a Kelvin wave to a seamount resonance problem.
- The plotting has been rewritten to take advantage of the new features of NCAR graphics version 3.0, especially those in **conpack**.
- Several changes to the plotting subroutines, including the addition of slices along surfaces of constant η.
- The calls to **mud2** have been modified for version 2.0 of the **mud2** elliptic solver.
- The reading and writing of the history file is now all in subroutine **histio**.
- The flag **slctpoly** has been taken out again until we are more comfortable with using alternate polynomials.
- The common blocks have been put in an include file and alphabetized. The SPEM can be obtained with either the include statements and the include file or with the common blocks already included.
- The code was run through a relabelling program.

1.4 Changes between version 1.0 and version 2.0

A number of minor changes have been made to the model, including:

- The timestepping has been changed from leapfrog-trapezoidal to a leapfrog with occasional Adams-Bashforth steps.
- A flag (**perchan**) has been added to allow the domain to be periodic in x.
- The plotting has been cleaned up and the plotting subroutines have all been put together.
- The model density, ρ, has been redefined so that $\rho_{\text{total}} = \rho_0 + \overline{\rho}(z) + \rho(x, y, z, t)$.
- The climatology fields **uclm**, **vclm**, and **rclm** have been added. The flags **uflags(9)**, **vflags(9)**, and **rflags(6)** switch the right-hand side terms which nudge the u, v and ρ fields towards the climatology on a timescale of $1/$**rdmp**.
- The subroutine **getgrid** now reads in h and f as well as the grid arrays. The program **ezgrid** writes out a rectangular grid with its topography and Coriolis parameter.
- A flag (**slctpoly**) has been added to allow the use of polynomials other than the Chebyshevs. These polynomials are calculated externally by a program which is available upon request and which will be documented in a future release.
- The variables **f0**, **beta**, **xh**, **yh**, **dx**, **dxsq**, **de**, and **desq** have all been deleted.

Chapter 2

Model Formulation

2.1 Equations of motion

The primitive equations in Cartesian coordinates can be written:

$$\frac{\partial u}{\partial t} + \vec{v} \cdot \nabla u - fv = -\frac{\partial \phi}{\partial x} + \mathcal{F}_u + \mathcal{D}_u \tag{2.1}$$

$$\frac{\partial v}{\partial t} + \vec{v} \cdot \nabla v + fu = -\frac{\partial \phi}{\partial y} + \mathcal{F}_v + \mathcal{D}_v \tag{2.2}$$

$$\frac{\partial T}{\partial t} + \vec{v} \cdot \nabla T = \mathcal{F}_T + \mathcal{D}_T \tag{2.3}$$

$$\frac{\partial S}{\partial t} + \vec{v} \cdot \nabla S = \mathcal{F}_S + \mathcal{D}_S \tag{2.4}$$

$$\rho = \rho(T, S, P) \tag{2.5}$$

$$\frac{\partial \phi}{\partial z} = \frac{-\rho g}{\rho_o} \tag{2.6}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{2.7}$$

where, in standard notation:

(u, v, w) = the (x, y, z) components of vector velocity $\vec{v}$

$\rho_o + \rho(x, y, z, t)$ = total density

$T(x, y, z, t)$ = total temperature

$S(x, y, z, t)$ = total salinity

P = total pressure $P \approx -\rho_o g z$

$\phi(x, y, z, t)$ = dynamic pressure (p/ρ_o)

$f(x, y)$ = Coriolis parameter

g = acceleration of gravity

$(\mathcal{F}_u, \mathcal{F}_v, \mathcal{F}_T, \mathcal{F}_S)$ = forcing terms

and

$(\mathcal{D}_u, \mathcal{D}_v, \mathcal{D}_T, \mathcal{D}_S)$ = diffusive terms.

Equations (2.1) and (2.2) express the momentum balance in the x and y directions, respectively. The time evolution of the perturbation temperature and salinity fields, $T(x,y,z,t)$ and $S(x,y,z,t)$, are governed by the advective-diffusive equations (2.3) and (2.4). The equation of state is given by equation (2.5). In the Boussinesq approximation, density variations are neglected in the momentum equations except in their contribution to the buoyancy force in the vertical momentum equation (2.6). Under the hydrostatic approximation, it is further assumed that the vertical pressure gradient balances the buoyancy force. Lastly, equation (2.7) expresses the continuity equation for an incompressible fluid. For the moment, the effects of forcing and dissipation will be represented by the schematic terms $\mathcal{F}$ and $\mathcal{D}$, respectively. The horizontal mixing is currently implemented as either harmonic (Laplacian) friction, biharmonic friction or a Shapiro filter (Shapiro [13]) along σ surfaces.

2.2 Vertical boundary conditions

The vertical boundary conditions can be prescribed as follows:

$$
\begin{aligned}
\text{top}(z=0) \qquad & \nu \tfrac{\partial u}{\partial z} = \tau_x^o(x,y,t) \\
& \nu \tfrac{\partial v}{\partial z} = \tau_y^o(x,y,t) \\
& \kappa_T \tfrac{\partial T}{\partial z} = \tau_T^o(x,y,t) \\
& \kappa_S \tfrac{\partial S}{\partial z} = \tau_S^o(x,y,t) \\
& w = 0 \\
\text{and bottom } (z=-h) \qquad & \nu \tfrac{\partial u}{\partial z} = \tau_x^h(x,y,t) \\
& \nu \tfrac{\partial v}{\partial z} = \tau_y^h(x,y,t) \\
& \kappa_T \tfrac{\partial T}{\partial z} = \tau_T^h(x,y,t) \\
& \kappa_S \tfrac{\partial S}{\partial z} = \tau_S^h(x,y,t) \\
& \vec{v} \cdot \nabla h = 0.
\end{aligned}
$$

Surface distributions of wind stress (τ_x^o, τ_y^o) and vertical density flux (τ_T^o, τ_S^o) are prescribed on the top. On the variable bottom, $z = -h(x,y)$, the horizontal velocity components are constrained to accommodate a prescribed bottom stress; the vertical heat and salt flux is also prescribed. It is relevant to note at this point that the vertical (polynomial) discretization schemes to be described in section 3.1 can accommodate quite arbitrary boundary conditions on $z = 0, -h$.

2.3 Horizontal boundary conditions

As distributed, the model configuration is that of a periodic channel. If the variable **perchan** is set to **false** then the boundary conditions are that of a closed basin. For any other configuration it is the responsibility of the user to provide the appropriate boundary conditions on u, v, T, S, and ψ, where ψ is the transport streamfunction. At every time step the subroutines **bcs** and **bcpsi** are called to fill in the necessary boundary values.

If the biharmonic friction is used, a higher order boundary condition must also be provided. The model currently has this built into the code where the biharmonic terms are calculated. The high order boundary conditions used are $\frac{\partial}{\partial x}\left(\frac{h\nu}{mn}\frac{\partial^2 u}{\partial x^2}\right) = 0$ on the eastern and western boundaries and $\frac{\partial}{\partial y}\left(\frac{h\nu}{mn}\frac{\partial^2 u}{\partial y^2}\right) = 0$ on the northern and southern boundaries. The boundary conditions for v, T, and S are similar. These boundary conditions were chosen because they preserve the property of no gain or loss of momentum, temperature, or salt across a closed boundary.

2.4 Sigma (stretched vertical) coordinate system

From the point of view of the computational model, it is highly convenient to introduce a stretched vertical coordinate system which essentially "flattens out" the variable bottom at $z = -h(x, y)$. Such "sigma" coordinate systems have long been used, with slight appropriate modification, in both meteorology and oceanography (e.g., Phillips [11] and Freeman et al. [7]). To proceed, we make the coordinate transformation:

$$\begin{aligned} \hat{x} &= x \\ \hat{y} &= y \\ \sigma &= 1 + 2(z/h) \end{aligned}$$

and

$$\hat{t} = t.$$

In the stretched system, the vertical coordinate σ spans the range $-1 \leq \sigma \leq 1$; we are therefore left with level upper ($\sigma = 1$) and lower ($\sigma = -1$) bounding surfaces. As a trade-off for this geometric simplification, the dynamic equations become somewhat more complicated. The resulting dynamic equations are, dropping the carats $\hat{}$ and remembering the chain rule $\left.\frac{\partial\phi}{\partial x}\right|_z = \left.\frac{\partial\phi}{\partial x}\right|_\sigma + (1-\sigma)\frac{h_x}{h}\frac{\partial\phi}{\partial\sigma}$:

$$\frac{\partial u}{\partial t} - fv + \vec{v}\cdot\nabla u = -\frac{\partial\phi}{\partial x} + (1-\sigma)\left(\frac{g\rho}{2\rho_o}\right)\frac{\partial h}{\partial x} + \mathcal{F}_u + \mathcal{D}_u \tag{2.8}$$

$$\frac{\partial v}{\partial t} + fu + \vec{v}\cdot\nabla v = -\frac{\partial\phi}{\partial y} + (1-\sigma)\left(\frac{g\rho}{2\rho_o}\right)\frac{\partial h}{\partial y} + \mathcal{F}_v + \mathcal{D}_v \tag{2.9}$$

$$\frac{\partial T}{\partial t} + \vec{v}\cdot\nabla T = \mathcal{F}_T + \mathcal{D}_T \tag{2.10}$$

$$\frac{\partial S}{\partial t} + \vec{v}\cdot\nabla S = \mathcal{F}_S + \mathcal{D}_S \tag{2.11}$$

$$\rho = \rho(T, S, P) \tag{2.12}$$

$$\frac{\partial\phi}{\partial\sigma} = \left(\frac{-gh\rho}{2\rho_o}\right) \tag{2.13}$$

$$\frac{\partial}{\partial x}(hu) + \frac{\partial}{\partial y}(hv) + h\frac{\partial\Omega}{\partial\sigma} = 0 \tag{2.14}$$

where

$$\vec{v} = (u, v, \Omega)$$

$$\vec{v} \cdot \nabla = u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + \Omega\frac{\partial}{\partial \sigma},$$

and the vertical velocity in sigma coordinates is

$$\Omega(x, y, \sigma, t) = \frac{1}{h}\left[(1-\sigma)u\frac{\partial h}{\partial x} + (1-\sigma)v\frac{\partial h}{\partial y} + 2w\right].$$

In the stretched coordinate, the vertical boundary conditions become:

$$\begin{aligned}
\text{top}(\sigma = 1) \qquad & \left(\tfrac{2\nu}{h}\right)\tfrac{\partial u}{\partial \sigma} = \tau_x^o(x, y, t) \\
& \left(\tfrac{2\nu}{h}\right)\tfrac{\partial v}{\partial \sigma} = \tau_y^o(x, y, t) \\
& \left(\tfrac{2\kappa_T}{h}\right)\tfrac{\partial T}{\partial \sigma} = \tau_T^o(x, y, t) \\
& \left(\tfrac{2\kappa_S}{h}\right)\tfrac{\partial S}{\partial \sigma} = \tau_S^o(x, y, t) \\
& \Omega = 0 \\
\text{and bottom } (\sigma = -1) \qquad & \left(\tfrac{2\nu}{h}\right)\tfrac{\partial u}{\partial \sigma} = \tau_x^h(x, y, t) \\
& \left(\tfrac{2\nu}{h}\right)\tfrac{\partial v}{\partial \sigma} = \tau_y^h(x, y, t) \\
& \left(\tfrac{2\kappa_T}{h}\right)\tfrac{\partial T}{\partial \sigma} = 0 \\
& \left(\tfrac{2\kappa_S}{h}\right)\tfrac{\partial S}{\partial \sigma} = 0 \\
& \Omega = 0.
\end{aligned}$$

Note the simplification of the lower (vertical velocity) boundary condition which arises from the sigma coordinate transformation.

2.5 Horizontal curvilinear coordinates

In many applications of interest (e.g., flow adjacent to a coastal boundary), the fluid may be confined horizontally within an irregular region. In such problems, a horizontal coordinate system which conforms to the irregular lateral boundaries is advantageous. It is often also true in many geophysical problems that the simulated flow fields have regions of enhanced structure (e.g., boundary currents or fronts) which occupy a relatively small fraction of the physical /computational domain. In these problems, added efficiency can be gained by placing more (computational) resolution in such regions.

The requirement for a boundary-following coordinate system and for a laterally variable grid resolution can both be met (for suitably smooth domains) by introducing an appropriate orthogonal coordinate transformation in the horizontal. Let the new coordinates be $\xi(x, y)$ and $\eta(x, y)$ where the relationship of horizontal arc length to the differential distance is given by:

$$(ds)_\xi = \left(\frac{1}{m}\right) d\xi \tag{2.15}$$

$$(ds)_\eta = \left(\frac{1}{n}\right) d\eta \tag{2.16}$$

Here, $m(\xi,\eta)$ and $n(\xi,\eta)$ are the scale factors which relate the differential distances $(\Delta\xi, \Delta\eta)$ to the actual (physical) arc lengths.

Denoting the velocity components in the new coordinate system by

$$\vec{v}\cdot\hat{\xi} = u \tag{2.17}$$

and

$$\vec{v}\cdot\hat{\eta} = v \tag{2.18}$$

the equations of motion (2.8)-(2.14) can be re-written (see, e.g., Arakawa and Lamb [4]):

$$\begin{aligned}&\frac{\partial}{\partial t}\left(\frac{hu}{mn}\right) + \frac{\partial}{\partial\xi}\left(\frac{hu^2}{n}\right) + \frac{\partial}{\partial\eta}\left(\frac{huv}{m}\right) + \frac{\partial}{\partial\sigma}\left(\frac{hu\Omega}{mn}\right)\\ &\quad - \left\{\left(\frac{f}{mn}\right) + v\frac{\partial}{\partial\xi}\left(\frac{1}{n}\right) - u\frac{\partial}{\partial\eta}\left(\frac{1}{m}\right)\right\} hv =\\ &\quad - \left(\frac{h}{n}\right)\frac{\partial\phi}{\partial\xi} + (1-\sigma)\left(\frac{gh\rho}{2\rho_o n}\right)\frac{\partial h}{\partial\xi} + \mathcal{F}_u + \mathcal{D}_u\end{aligned} \tag{2.19}$$

$$\begin{aligned}&\frac{\partial}{\partial t}\left(\frac{hv}{mn}\right) + \frac{\partial}{\partial\xi}\left(\frac{huv}{n}\right) + \frac{\partial}{\partial\eta}\left(\frac{hv^2}{m}\right) + \frac{\partial}{\partial\sigma}\left(\frac{hv\Omega}{mn}\right)\\ &\quad + \left\{\left(\frac{f}{mn}\right) + v\frac{\partial}{\partial\xi}\left(\frac{1}{n}\right) - u\frac{\partial}{\partial\eta}\left(\frac{1}{m}\right)\right\} hu =\\ &\quad - \left(\frac{h}{m}\right)\frac{\partial\phi}{\partial\eta} + (1-\sigma)\left(\frac{gh\rho}{2\rho_o m}\right)\frac{\partial h}{\partial\eta} + \mathcal{F}_v + \mathcal{D}_v\end{aligned} \tag{2.20}$$

$$\frac{\partial}{\partial t}\left(\frac{hT}{mn}\right) + \frac{\partial}{\partial\xi}\left(\frac{huT}{n}\right) + \frac{\partial}{\partial\eta}\left(\frac{hvT}{m}\right) + \frac{\partial}{\partial\sigma}\left(\frac{h\Omega T}{mn}\right) = \mathcal{F}_T + \mathcal{D}_T \tag{2.21}$$

$$\frac{\partial}{\partial t}\left(\frac{hS}{mn}\right) + \frac{\partial}{\partial\xi}\left(\frac{huS}{n}\right) + \frac{\partial}{\partial\eta}\left(\frac{hvS}{m}\right) + \frac{\partial}{\partial\sigma}\left(\frac{h\Omega S}{mn}\right) = \mathcal{F}_S + \mathcal{D}_S \tag{2.22}$$

$$\rho = \rho(T,S,P) \tag{2.23}$$

$$\frac{\partial\phi}{\partial\sigma} = -\left(\frac{gh\rho}{2\rho_o}\right) \tag{2.24}$$

$$\frac{\partial}{\partial\xi}\left(\frac{hu}{n}\right) + \frac{\partial}{\partial\eta}\left(\frac{hv}{m}\right) + \frac{\partial}{\partial\sigma}\left(\frac{h\Omega}{mn}\right) = 0. \tag{2.25}$$

All boundary conditions remain unchanged.

Chapter 3

Numerical Solution Technique

3.1 Vertical spectral method

The vertical (σ) dependence of the model variables is represented as an expansion in a finite polynomial basis set $P_k(\sigma)$; that is, we set

$$b(\xi,\eta,\sigma) = \sum_{k=0}^{N} P_k(\sigma)\hat{b}_k(\xi,\eta) \tag{3.1}$$

where the arbitrary variable b, introduced here for convenience, is any of u, v, ρ, T, S, ϕ, or Ω. The only restriction placed on the form of the basis polynomials is that $\frac{1}{2}\int_{-1}^{1} P_k(\sigma)d\sigma = \delta_{0k}$, i.e., only the lowest order polynomial may have a non-zero vertical integral. This isolates the external mode (or depth averaged component of the field) in the amplitude of the $k = 0$ polynomial. In practice, the spectral technique does not explicitly solve for the polynomial coefficients $\hat{b}_k$ but rather for the actual variable values at "collocation" points (or equivalent grid points) σ_n chosen to correspond to the location of the extrema of the highest order polynomial. The collocation point values b_n are thus defined as

$$b_n = b(\sigma_n) = \sum_{k=0}^{N} P_k(\sigma_n)\,\hat{b}_k \qquad 0 \le n \le N \tag{3.2}$$

and will be functions of (ξ,η). The polynomial coefficients $\hat{b}_k$ can be recovered from the collocation point values b_n by a linear matrix transformation. Consider a matrix F whose elements are

$$F_{nk} = P_k(\sigma_n) \tag{3.3}$$

and let b and $\hat{b}$ be column vectors whose elements are b_n and $\hat{b}_k$ respectively. Then

$$b = F\hat{b} \tag{3.4}$$

and hence

$$\hat{b} = F^{-1}b. \tag{3.5}$$

It is now straightforward to represent vertical differentiation and integration in terms of matrix operators. Consider a matrix R whose elements are

$$R_{nk} = \frac{\partial P_k(\sigma_n)}{\partial\sigma}. \tag{3.6}$$

Then the matrix $C_{DZ} = RF^{-1}$ will perform differentiation of the model field (b) in the vertical direction, denoted in the following subsections by the δ_σ operation, since

$$\delta_\sigma b = \frac{\partial P_k(\sigma_n)}{\partial\sigma}\hat{b}_k = R\hat{b} = RF^{-1}b = C_{DZ}b. \tag{3.7}$$

Similarly, consider a matrix S whose elements are

$$S_{nk} = \int_{\sigma_n}^{1} P_k(\sigma)\, d\sigma. \tag{3.8}$$

Then the matrix $C_{INT} = SF^{-1}$ will perform vertical integration, denoted in the following sections by the I_σ^1 operation, since

$$I_\sigma^1 b = \int_\sigma^1 b\, d\sigma = \sum_{k=0}^{N} \int_{\sigma_n}^{1} P_k(\sigma)\, d\sigma\, \hat{b}_k = S\hat{b} = SF^{-1}b = C_{INT}b. \tag{3.9}$$

By default, the SPEM uses the Chebyshev polynomials. The matrix operators F, R and S and collocation levels σ_n for this basis set are given in Appendix B.

3.2 The collocation method

The upper and lower boundary conditions can be conveniently implemented for the polynomials using the collocation method. The boundary conditions are of the form where the vertical derivatives of the fields are known, for instance

$$\nu \frac{\partial u}{\partial z} = \tau_x^o.$$

This comes into the dynamical equations in the term

$$\frac{\partial}{\partial z}\left(\nu \frac{\partial u}{\partial z}\right).$$

The vertical derivative of u is calculated using the operator C_{DZ} (equation (3.7)) and then the values of $\nu \frac{\partial u}{\partial z}$ at $\sigma = \pm 1$ are replaced with the known values prior to the second vertical derivative being taken. This procedure with the rules for vertical integration is guaranteed to preserve the global balance of momentum, heat, and salt.

The form of boundary conditions where the boundary values of u (or v, T or S) are known can also be implemented using the collocation method. In that case the dynamical equations for u on the boundary levels are replaced by setting the value of u equal to the known value. This approach is quite commonly used in finite-difference techniques, and is extremely convenient when the boundary conditions provide boundary values for the function being sought.

3.3 Horizontal grid system

In the horizontal (ξ, η), a traditional, centered, second-order finite-difference approximation is adopted. In particular, the horizontal arrangement of variables is as shown in figure 3.1. This is equivalent to the well-known Arakawa "C" grid, which is well suited for problems with horizontal resolution that is fine compared to the first radius of deformation (Arakawa and Lamb [4]).

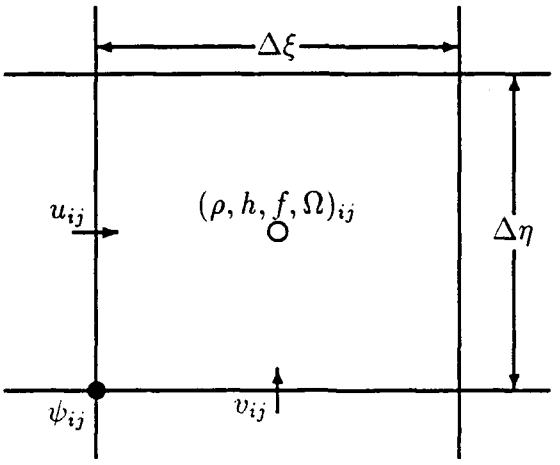


Figure 3.1: Placement of variables on an Arakawa C grid

3.4 Conservation properties

SPEM conserves the first and (optionally) the second moments of u, v, S, and T, where the second moments of velocity correspond to kinetic energy.

3.4.1 First moment conservation

To conserve first moments, the four vertical advection terms

$$\frac{\partial}{\partial\sigma}\left(\frac{h\Omega u}{mn}\right), \frac{\partial}{\partial\sigma}\left(\frac{h\Omega v}{mn}\right), \frac{\partial}{\partial\sigma}\left(\frac{h\Omega S}{mn}\right) \text{ and } \frac{\partial}{\partial\sigma}\left(\frac{h\Omega T}{mn}\right)$$

are computed as follows:

$$\frac{\partial}{\partial\sigma}\left(\frac{h\Omega q}{mn}\right) = \left(\frac{h\Omega}{mn}\right)\frac{\partial q}{\partial\sigma} + \left(\frac{hq}{mn}\right)\frac{\partial\Omega}{\partial\sigma} \equiv \mathcal{A}_q$$

where q denotes any of the four quantities u, v, S and T.

To ensure the conservation of the first moment the computed advection term is vertically integrated

$$\mathcal{A}_{qBT} = \frac{1}{2}\int_{-1}^{+1} \mathcal{A}_q \, d\sigma$$

and the vertical advection term is corrected by

$$\mathcal{A}_{qBC} = \mathcal{A}_q - \mathcal{A}_{qBT}.$$

This local conservation property was implemented in version 1.0 of SPEM.

3.4.2 Second moment conservation

Version 3.0 will also conserve second moments in a global sense, if desired. To ensure this conservation of a quantity q, the following integral must vanish (this can be seen when each term is integrated by parts and the boundary conditions $\Omega = 0$ at $\sigma = 1, -1$ are used):

$$Q_D = \int_{-1}^{+1} \left(2q\mathcal{A}_{qBC} - \left(\frac{hq^2}{mn} \right) \frac{\partial \Omega}{\partial \sigma} \right) d\sigma$$

where q denotes u, v, S or T. The horizontal integral of these deviations from zero

$$\mathcal{S}_Q = \iint Q_D \, d\xi \, d\eta$$

is projected onto the existing q field using

$$\mathcal{S}_q = \iiint_{-1}^{+1} (q \cdot q_{BC}) \, d\sigma \, d\xi \, d\eta$$

where q_{BC} is the baroclinic part of q ($q_{BC} = q - \frac{1}{2}\int_{-1}^{+1} q \, d\sigma$). The correction

$$\mathcal{A}^*_{qBC} = \mathcal{A}_{qBC} + \frac{\mathcal{S}_Q}{\mathcal{S}_q} q_{BC}$$

leads to the global conservation of quadratic moments (i.e. kinetic energy, T^2, and S^2).

3.5 Semi-discrete equations

Using the representations described in sections 3.1 and 3.3, the semi-discrete form of the dynamic equations (2.19)–(2.25) is:

$$\frac{\partial}{\partial t}\left(\frac{u\overline{h}^{\xi}}{\overline{m}^{\xi}\overline{n}^{\xi}} \right) + \delta_\xi \left\{ \overline{u}^{\xi} \overline{\left(\frac{u\overline{h}^{\xi}}{\overline{n}^{\xi}} \right)}^{\xi} \right\} + \delta_\eta \left\{ \overline{u}^{\eta} \overline{\left(\frac{v\overline{h}^{\eta}}{\overline{m}^{\eta}} \right)}^{\xi} \right\} + \delta_\sigma \left\{ \frac{u\overline{h}^{\xi}\overline{\Omega}^{\xi}}{\overline{m}^{\xi}\overline{n}^{\xi}} \right\}$$

$$- \overline{\left\{ \frac{\overline{f}^{\xi\eta}\overline{h}^{\xi\eta}}{\overline{m}^{\xi\eta}\overline{n}^{\xi\eta}} + \overline{v}^{\xi} \overline{h}^{\xi\eta} \overline{\delta_\xi \left(\frac{1}{n} \right)}^{\xi\eta} - \overline{u}^{\eta} \overline{h}^{\xi\eta} \overline{\delta_\eta \left(\frac{1}{m} \right)}^{\xi\eta} \right\} \overline{v}^{\xi}}^{\eta} =$$

$$- \frac{\overline{h}^{\xi}}{\overline{n}^{\xi}} \delta_\xi \phi + (1-\sigma) \frac{g\overline{h}^{\xi}\overline{\rho}^{\xi}}{2\rho_o \overline{n}^{\xi}} \delta_\xi h + \mathcal{D}_u + \mathcal{F}_u \qquad (3.10)$$

$$\frac{\partial}{\partial t}\left(\frac{v\overline{h}^{\eta}}{\overline{m}^{\eta}\overline{n}^{\eta}} \right) + \delta_\xi \left\{ \overline{v}^{\xi} \overline{\left(\frac{u\overline{h}^{\xi}}{\overline{n}^{\xi}} \right)}^{\eta} \right\} + \delta_\eta \left\{ \overline{v}^{\eta} \overline{\left(\frac{v\overline{h}^{\eta}}{\overline{m}^{\eta}} \right)}^{\eta} \right\} + \delta_\sigma \left\{ \frac{v\overline{h}^{\eta}\overline{\Omega}^{\eta}}{\overline{m}^{\eta}\overline{n}^{\eta}} \right\}$$

$$+ \overline{\left\{ \frac{\overline{f}^{\xi\eta}\overline{h}^{\xi\eta}}{\overline{m}^{\xi\eta}\overline{n}^{\xi\eta}} + \overline{v}^{\xi} \overline{h}^{\xi\eta} \overline{\delta_\xi \left(\frac{1}{n} \right)}^{\xi\eta} - \overline{u}^{\eta} \overline{h}^{\xi\eta} \overline{\delta_\eta \left(\frac{1}{m} \right)}^{\xi\eta} \right\} \overline{u}^{\eta}}^{\xi} =$$

$$- \frac{\overline{h}^{\eta}}{\overline{m}^{\eta}} \delta_\eta \phi + (1-\sigma) \frac{g\overline{h}^{\eta}\overline{\rho}^{\eta}}{2\rho_o \overline{m}^{\eta}} \delta_\eta h + \mathcal{D}_v + \mathcal{F}_v \qquad (3.11)$$

$$\frac{\partial}{\partial t}\left(\frac{hT}{mn}\right) + \delta_\xi\left\{\frac{u\overline{h}^\xi \overline{T}^\xi}{\overline{n}^\xi}\right\} + \delta_\eta\left\{\frac{v\overline{h}^\eta \overline{T}^\eta}{\overline{m}^\eta}\right\} + \delta_\sigma\left\{\frac{h\Omega T}{mn}\right\} = \mathcal{D}_T + \mathcal{F}_T \tag{3.12}$$

$$\frac{\partial}{\partial t}\left(\frac{hS}{mn}\right) + \delta_\xi\left\{\frac{u\overline{h}^\xi \overline{S}^\xi}{\overline{n}^\xi}\right\} + \delta_\eta\left\{\frac{v\overline{h}^\eta \overline{S}^\eta}{\overline{m}^\eta}\right\} + \delta_\sigma\left\{\frac{h\Omega S}{mn}\right\} = \mathcal{D}_S + \mathcal{F}_S \tag{3.13}$$

$$\rho = \rho(S, T, P) \tag{3.14}$$

$$\phi(\sigma) = -\left(\frac{gh}{2\rho_o}\right) I_\sigma^1 \rho \tag{3.15}$$

$$\delta_\xi\left\{\frac{u\overline{h}^\xi}{\overline{n}^\xi}\right\} + \delta_\eta\left\{\frac{v\overline{h}^\eta}{\overline{m}^\eta}\right\} + \delta_\sigma\left\{\frac{h\Omega}{mn}\right\} = 0. \tag{3.16}$$

Here δ_ξ and δ_η denote simple centered finite-difference approximations to $\partial/\partial\xi$ and $\partial/\partial\eta$, with the differences taken over the distances $\Delta\xi$ and $\Delta\eta$, respectively; $\overline{(\ \ )}^\xi$ and $\overline{(\ \ )}^\eta$ represent averages taken over the distances $\Delta\xi$ and $\Delta\eta$; δ_σ represents a vertical derivative evaluated according to the prescription described in Section 3.1; and I_σ^1 indicates the analogous vertical integral. The vertical advection terms are actually more complicated, as described in section 3.4.

This method of averaging was chosen because it internally conserves mass and energy in the model domain, although it is still possible to exchange mass and energy through the open boundaries. The method is similar to that used in Arakawa and Lamb [4]; however, their scheme also conserves enstrophy.

3.6 Time-stepping: depth-integrated flow ($k = 0$ mode)

In continuous form, the horizontal momentum and continuity equations (2.19), (2.20) and (2.25) can be written:

$$\frac{\partial}{\partial t}\left(\frac{hu}{mn}\right) = R_u - \left(\frac{h}{n}\right)\frac{\partial \phi}{\partial \xi} \tag{3.17}$$

$$\frac{\partial}{\partial t}\left(\frac{hv}{mn}\right) = R_v - \left(\frac{h}{m}\right)\frac{\partial \phi}{\partial \eta} \tag{3.18}$$

and

$$\frac{\partial}{\partial \xi}\left(\frac{hu}{n}\right) + \frac{\partial}{\partial \eta}\left(\frac{hv}{m}\right) + \frac{\partial}{\partial \sigma}\left(\frac{h\Omega}{mn}\right) = 0, \tag{3.19}$$

where R_u and R_v represent the sum of all other terms in the u and v equations, respectively. Performing a vertical average

$$\left(\frac{1}{2}\int_{-1}^{1} d\sigma\right),$$

the equations become:

$$\frac{\partial}{\partial t}\left(\frac{h\overline{u}}{mn}\right) = \overline{R_u} - \left(\frac{h}{n}\right)\frac{\partial\overline{\phi}}{\partial\xi} \tag{3.20}$$

$$\frac{\partial}{\partial t}\left(\frac{h\overline{v}}{mn}\right) = \overline{R_v} - \left(\frac{h}{m}\right)\frac{\partial\overline{\phi}}{\partial\eta} \tag{3.21}$$

and

$$\frac{\partial}{\partial\xi}\left(\frac{h\overline{u}}{n}\right) + \frac{\partial}{\partial\eta}\left(\frac{h\overline{v}}{m}\right) = 0, \tag{3.22}$$

where the overbar indicates a vertically averaged quantity.

By virtue of (3.22), the depth-averaged flow is horizontally non-divergent. This enables us to represent it in terms of a streamfunction, $\psi(\xi,\eta,t)$, such that

$$\overline{u} = -\left(\frac{n}{h}\right)\frac{\partial\psi}{\partial\eta} \tag{3.23}$$

and

$$\overline{v} = \left(\frac{m}{h}\right)\frac{\partial\psi}{\partial\xi} \tag{3.24}$$

By re-arrangement:

$$\frac{\partial}{\partial t}\left(\frac{\overline{u}}{m}\right) = \left(\frac{n}{h}\right)\overline{R_u} - \frac{\partial\overline{\phi}}{\partial\xi}$$

and

$$\frac{\partial}{\partial t}\left(\frac{\overline{v}}{n}\right) = \left(\frac{m}{h}\right)\overline{R_v} - \frac{\partial\overline{\phi}}{\partial\eta}.$$

Taking the curl of these equations yields a vorticity equation for the depth-averaged flow:

$$\frac{\partial q}{\partial t} = \frac{\partial}{\partial t}\left\{\frac{\partial}{\partial\xi}\left(\frac{\overline{v}}{n}\right) - \frac{\partial}{\partial\eta}\left(\frac{\overline{u}}{m}\right)\right\} = \frac{\partial}{\partial\xi}\left(\frac{m}{h}\overline{R_v}\right) - \frac{\partial}{\partial\eta}\left(\frac{n}{h}\overline{R_u}\right) = R_q \tag{3.25}$$

or, using (3.23) and (3.24)

$$\frac{\partial}{\partial t}\left\{\left(\frac{m}{hn}\right)\frac{\partial^2\psi}{\partial\xi^2} + \left(\frac{n}{hm}\right)\frac{\partial^2\psi}{\partial\eta^2} + \frac{\partial}{\partial\xi}\left(\frac{m}{hn}\right)\frac{\partial\psi}{\partial\xi} + \frac{\partial}{\partial\eta}\left(\frac{n}{hm}\right)\frac{\partial\psi}{\partial\eta}\right\} = R_q. \tag{3.26}$$

The introduction of the streamfunction ψ serves two purposes. First, it automatically guarantees horizontal non-divergence of the transport field, as required by (3.22). Second, it eliminates any dependence on the depth-averaged component of the pressure field ϕ which contains an unknown contribution arising due to the rigid lid.

The vorticity equation (3.26) is solved by first obtaining an updated value of q by application of the leapfrog (second-order) time-differencing scheme described in Appendix A. The associated value of the streamfunction is determined from the generalized elliptic equation:

$$\left(\frac{m}{hn}\right)\frac{\partial^2\psi}{\partial\xi^2} + \left(\frac{n}{hm}\right)\frac{\partial^2\psi}{\partial\eta^2} + \frac{\partial}{\partial\xi}\left(\frac{m}{hn}\right)\frac{\partial\psi}{\partial\xi} + \frac{\partial}{\partial\eta}\left(\frac{n}{hm}\right)\frac{\partial\psi}{\partial\eta} = q. \tag{3.27}$$

A solution to (3.27) is fully prescribed by specifying the values of ψ on the boundary of the model domain. The SPEM currently uses the NCAR elliptic solver **mud2** (see Adams [2]) to solve equation (3.27).

3.7 Time-stepping: internal velocity modes and density field

The N internal modes of the velocity distribution $[(u,v)_{ijk}$ for $1 \leq k \leq N]$ are obtained by direct time-stepping of equations (3.10) and(3.11), having removed their depth-averaged component. The temperature and salinity equations (3.12) and (3.13) can be similarly advanced for $0 \leq k \leq N$. See Appendix A for a description of the time-stepping algorithms.

3.8 Determination of the vertical velocity and density fields

Having obtained a complete specification of the u, v, T, and S fields at the next time level by the methods outlined above, the vertical velocity and density fields can be calculated. The vertical velocity is obtained by vertical integration (as prescribed in section 3.1): in particular:

$$\Omega(\sigma) = \left(\frac{mn}{h}\right) I_\sigma^1 \left\{ \delta_\xi \left(\frac{u\overline{h}^\xi}{\overline{n}^\xi} \right) + \delta_\eta \left(\frac{v\overline{h}^\eta}{\overline{m}^\eta} \right) \right\}. \tag{3.28}$$

The density is obtained from the temperature and salinity via an equation of state. The SPEM provides a choice of a nonlinear equation of state $\rho = \rho(T, S, z)$ or a linear equation of state $\rho = \rho(T)$. The user is free to implement their preferred equation of state.

3.9 A corrected formulation of the pressure gradient terms

The pressure gradient terms in equations (2.19) and (2.20) are written in the form

$$h\nabla\phi + (1-\sigma)h\frac{\partial}{\partial\sigma}\left(\frac{\phi}{h}\right)\nabla h$$

or

$$h\nabla\phi + (1-\sigma)\left(\frac{gh\rho}{2\rho_o}\right)\nabla h.$$

This is the form traditionally used in sigma-coordinate models to account for the horizontal differences being taken along surfaces of constant σ. This form can be shown to lead to significant errors when $|\nabla h|$ is large (Beckmann, personal communication).

In an effort to reduce these errors the SPEM will optionally calculate additional correction terms. These terms are based on a Taylor series of the pressure at the ρ points on either side of a velocity point (where the pressure gradient is needed). See figure 3.2 for the geometry of the problem.

The pressure values which are required are $\phi_1(z_n - \Delta z)$ and $\phi_2(z_n + \Delta z)$ where $\Delta z = 1/4(\sigma - 1)\Delta h$. This can be expanded in z coordinates as:

$$\phi_1(z_n - \Delta z) = \phi_1(z_n) - \Delta z \frac{\partial \phi_1(z_n)}{\partial z} +$$

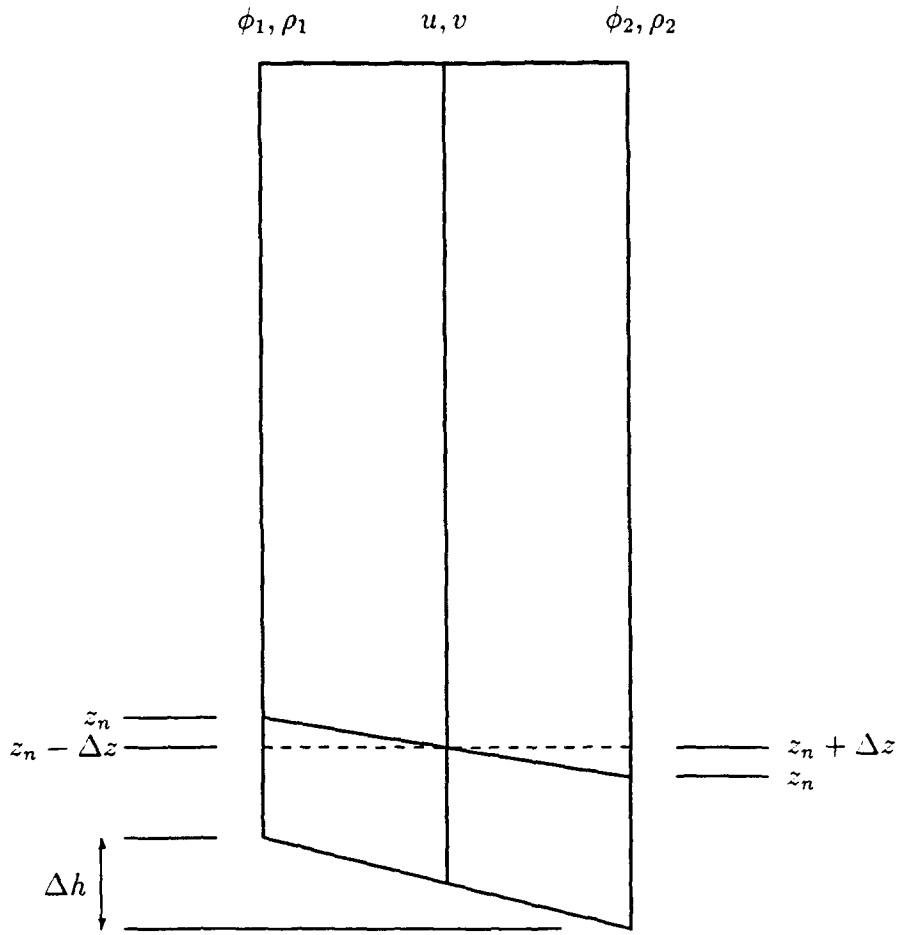


Figure 3.2: Geometry of the pressure gradient calculation

$$\frac{1}{2}(\Delta z)^2\frac{\partial^2\phi_1(z_n)}{\partial z^2} - \frac{1}{6}(\Delta z)^3\frac{\partial^3\phi_1(z_n)}{\partial z^3} + \cdots$$

$$\phi_2(z_n + \Delta z) = \phi_2(z_n) + \Delta z\frac{\partial\phi_2(z_n)}{\partial z} +$$

$$\frac{1}{2}(\Delta z)^2\frac{\partial^2\phi_2(z_n)}{\partial z^2} + \frac{1}{6}(\Delta z)^3\frac{\partial^3\phi_2(z_n)}{\partial z^3} + \cdots$$

This can be transformed to σ coordinates by:

$$\phi_1(z_n - \Delta z) = \phi_1(\sigma_n) - \Delta z\left(\frac{2}{h_1}\right)\frac{\partial\phi_1(\sigma_n)}{\partial\sigma} +$$

$$\frac{1}{2}(\Delta z)^2\left(\frac{2}{h_1}\right)^2\frac{\partial^2\phi_1(\sigma_n)}{\partial\sigma^2} - \cdots$$

$$\phi_2(z_n + \Delta z) = \phi_2(\sigma_n) + \Delta z\left(\frac{2}{h_2}\right)\frac{\partial\phi_2(\sigma_n)}{\partial\sigma} +$$

$$\frac{1}{2}(\Delta z)^2\left(\frac{2}{h_2}\right)^2\frac{\partial^2\phi_2(\sigma_n)}{\partial\sigma^2} + \cdots$$

The horizontal difference of these terms leads to the following form of the pressure gradient:

$$\frac{h}{n}\frac{\partial\phi}{\partial\xi}\bigg|_z = \frac{h}{n}\frac{\partial\phi}{\partial\xi}\bigg|_\sigma + \frac{1}{n}(1-\sigma)\phi_\sigma\frac{\partial h}{\partial\xi} +$$

$$\frac{h}{n}\left[\frac{1}{2}(1-\sigma)\frac{\partial h}{\partial\xi}\right]^2\left[\frac{1}{2}\frac{\partial}{\partial\xi}\left(\frac{\phi_{\sigma\sigma}}{h^2}\right) + \frac{1}{6}(1-\sigma)\frac{\phi_{\sigma\sigma\sigma}}{h^3}\frac{\partial h}{\partial\xi}\right] +$$

$$\frac{h}{n}\left[\frac{1}{2}(1-\sigma)\frac{\partial h}{\partial\xi}\right]^4\left[\frac{1}{24}\frac{\partial}{\partial\xi}\left(\frac{\phi_{\sigma\sigma\sigma\sigma}}{h^4}\right) + \frac{1}{120}(1-\sigma)\frac{\phi_{\sigma\sigma\sigma\sigma\sigma}}{h^5}\frac{\partial h}{\partial\xi}\right] + \cdots$$

and

$$\frac{h}{m}\frac{\partial\phi}{\partial\eta}\bigg|_z = \frac{h}{m}\frac{\partial\phi}{\partial\eta}\bigg|_\sigma + \frac{1}{m}(1-\sigma)\phi_\sigma\frac{\partial h}{\partial\eta} +$$

$$\frac{h}{m}\left[\frac{1}{2}(1-\sigma)\frac{\partial h}{\partial\eta}\right]^2\left[\frac{1}{2}\frac{\partial}{\partial\eta}\left(\frac{\phi_{\sigma\sigma}}{h^2}\right) + \frac{1}{6}(1-\sigma)\frac{\phi_{\sigma\sigma\sigma}}{h^3}\frac{\partial h}{\partial\eta}\right] +$$

$$\frac{h}{m}\left[\frac{1}{2}(1-\sigma)\frac{\partial h}{\partial\eta}\right]^4\left[\frac{1}{24}\frac{\partial}{\partial\eta}\left(\frac{\varphi_{\sigma\sigma\sigma\sigma}}{h^4}\right) + \frac{1}{120}(1-\sigma)\frac{\phi_{\sigma\sigma\sigma\sigma\sigma}}{h^5}\frac{\partial h}{\partial\eta}\right] + \cdots$$

This is a z-based pressure gradient which uses the vertical differentiation operator (see section 3.1) to extrapolate to the depth of the velocity collocation points. If both ϕ points are within the water column and $\mathbf{N} - 1$ correction terms are used, this calculation is equivalent to summing the modes to find ϕ at the given depth. Otherwise, we are using the model polynomials to extrapolate into the bottom.

3.10 Computation of the surface pressure

In timestepping the model from t to $t+\Delta t$, most of the terms in the momentum equations are computed directly. However, since the SPEM has a rigid lid, the effects of the surface pressure gradient are not immediately known. We get around having to calculate the surface pressure directly by calculating the curl of the momentum equations and finding the depth-integrated streamfunction as described in section 3.6. However, once the timestep is complete, it is possible to go back and infer what the surface pressure gradient should have been.

3.11 Convective adjustment

If desired, the density field is checked for occurences of static instability which the model will then attempt to remove. It cycles through the water column three times using a simple mixing scheme which conserves T and S. In the case of a nonlinear equation of state or of very strong advection this does not necessarily guarantee that the resulting density profile will be stable. The variables T, S and ρ are vertically mixed, but the corresponding velocity fields remain unchanged.

3.12 A comment on timing and vectorization

The SPEM is written in highly portable Fortran 77 and has been run on a number of different computers. A version with 42×41 gridpoints and 7 polynomials requires the amount of cpu time per timestep shown in table 3.1. This version has no diffusion of temperature or salinity and does not conserve second moments; turning options such as these back on will increase the execution time.

Computer	Time	Precision
Cray Y-MP	.083 sec	64 bits
Cray X-MP	.12 sec	64 bits
Cray 2	.17 sec	64 bits
Stardent Titan	3.6 sec	64 bits
SGI 4D/220 (IRIS)	2.3 sec	32 bits
Alliant FX-80	3.7 sec	32 bits
Alliant FX-80	6.4 sec	64 bits
Sun SparcStation 1+	5.5 sec	32 bits
VAX 8800	7.2 sec	32 bits

Table 3.1: Relative timings for several different computers. Although some of these machines have multiple processors, all timings were done on one processor

The model as it stands has been optimized for the Cray and uses the Cray versions of the routines **saxpy** and **sdot** for its matrix multiplication and dot products. If one were to run the model extensively on another type of machine it may be worth the trouble to configure the code for that machine. For instance, the Alliant has a compiler directive to tell it that a subroutine call within a loop can be split up among the processors. These compiler directives can be added where the subroutine call is independent for each loop index. (However, the most obvious of these loops have been restructured for better vectorization and are no longer independent.)

On supercomputers such as the Cray X-MP, efficient operation of models is closely tied to the degree to which codes can be vectorized. In the present SPEM, it is important to realize that, although vertical operations such as integration and differentiation are being carried out by matrix multiplication (slow transform) methods, such operations are nonetheless highly vectorizable when being simultaneously applied at a large number of adjacent horizontal points. In a case where there is a total of (say) $\mathcal{M}$ horizontal gridpoints, at each of which a vertical operation such as integration is needed, the requisite $\mathcal{M}$ integrations can be obtained entirely via manipulations of vectors of length $\mathcal{M}$. Since in most applications $\mathcal{M}$ will be many hundreds or thousands, and will typically be orders of magnitude greater than the number of vertical expansion functions ($\mathbf{N}$), vectorization of vertical operations by horizontal gridpoint is extremely powerful. This model takes full advantage of this vectorization technique.

Chapter 4

Details of the Code

4.1 Main subroutines

A flow chart for the main program is shown in figure 4.1. The boxes refer to subroutines which are described as follows:

bcs Horizontal boundary conditions are imposed upon the u, v, T, S and ψ fields. This can be modified by the user to provide solid boundaries or open boundaries of your favorite type. If the **perchan** switch is **true** then the boundaries are periodic in ξ.

init Does everything that needs to be done to start up the model run. It reads initial parameters and u, v, T, S, and ψ fields from disk or calls **peminit**. It then calculates the remaining initial fields and sends out the initial plots and opens the restart file. The flow chart for **init** is shown in figure 4.2.

omega Calculates the vertical velocity.

prsgrd Calculates the baroclinic pressure gradients, with an entry for calculating surface pressure gradients.

rhocal Calculates ρ using the equation of state.

srhs Calculates, stores, and tabulates contributions to the right hand side of equation (2.22), where the advective terms have been moved to the right hand side.

tmstp Performs the time step on T, S and the time step on the 'baroclinic part' of u and v via entry points **tstp**, **sstp**, **ustp**, and **vstp** respectively. The entry **vrtstp** performs the time step on ζ, solves for ψ, and completes the time step on u and v.

trhs Calculates, stores, and tabulates contributions to the right hand side of equation (2.21), where the advective terms have been moved to the right hand side.

urhs Calculates, stores, and tabulates contributions to the right hand side of equation (2.19), where all the terms other than $\frac{\partial}{\partial t}(\frac{hu}{mn})$ have been moved to the right hand side.

vrhs Calculates, stores, and tabulates contributions to the right hand side of equation (2.20), where all the terms other than $\frac{\partial}{\partial t}(\frac{hv}{mn})$ have been moved to the right hand side.

vrtrhs Calculates the R_q term in equation (3.26).

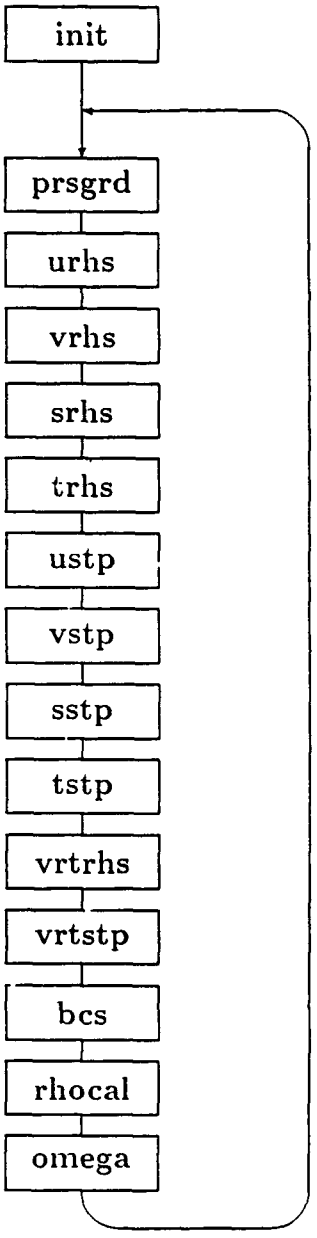


Figure 4.1: Flow chart for the model

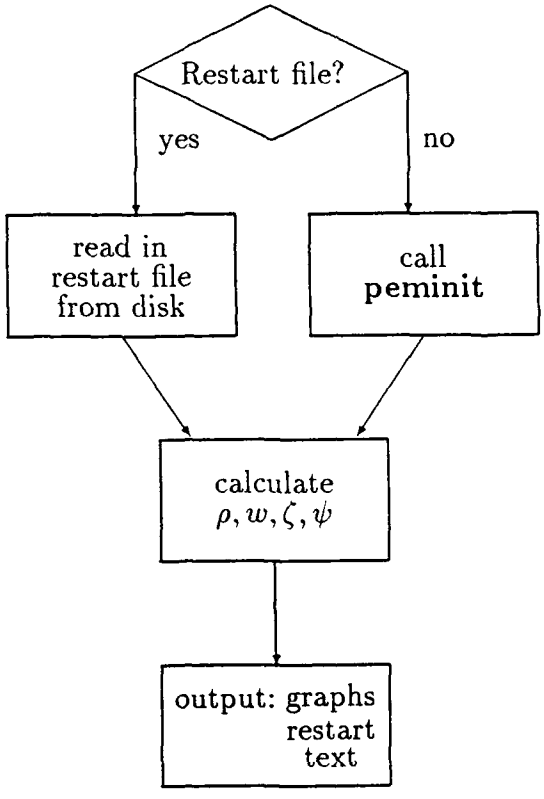


Figure 4.2: Flow chart for **init**

4.2 Other subroutines and functions

Initialization

chbset Sets up Chebyshev polynomial functions.

getgrid Reads in the curvilinear coordinate arrays as well as **f** and **h** from a grid-generation program.

histio Reads or writes the history file header.

- **iput** Reads model fields from a restart record on disk.
- **oput** Writes model fields to a restart record on disk.
- **spoput** Writes model surface pressure gradients to a restart record on disk. Separate from **oput** so that the initial surface pressure gradient can be calculated from the first timestep.

invmtx Matrix inversion — used only by **chbset**.

peminit Sets environmental parameters at $t = 0$.

Numerical techniques

bndyc Needed for **mud2** but never used.

coef Specifies the coefficients to the derivatives in **mud2**.

dmean Does a vectorized vertical integration of a model field and optionally subtracts the integral (mean) from the field.

mud2 Generalized elliptic solver (vectorized for Cray computers; Adams [2] and Adams [3]). Note: **mud2** only needs vorticity values in the interior and on boundaries for which the streamfunction values are not defined.

rstmix Mixes vertically in regions of static instability

saxpy Multiplies a constant by a vector and adds another vector. Used in multiplying matrices.

scopy Copies a vector to another vector.

sdot Computes an inner product of two vectors.

shapd1 Applies a one-dimensional Shapiro filter to a vector.

shapd2 Applies a two-dimensional Shapiro filter to a matrix by calling **shapd1** in each direction.

sscal Scales a vector by a constant scalar.

ssum Sums the elements of a vector.

Plotting Warning: for the slice plots an array which is dimensioned $\mathbf{L} \times \mathbf{M} \times 0 : \mathbf{N}$ is equivalenced to arrays which are $\mathbf{L} \times 51$ and $\mathbf{M} \times 51$. This will lead to problems for some values of **L**, **M**, and **N**. Fix this by commenting out the equivalences in the subroutines **pltslca** and **pltslcb**.

botplt Plots the bottom topography in slice plots.

cpmpxy Replacement for the **conpack** mapping routine so that the slice and slab contour plots come out in the appropriate coordinate system.

cpshift Makes **conpack** calls in such a way as to emulate our old modified **conrec**. It will optionally shift the contour lines by half a contour interval. See the version 3.0 NCAR graphics manual [5].

fill Fills an array for plotting horizontal sections of the model fields — either a level of constant sigma or the amplitude of a vertical mode.

fx, fy Used by **velvct** to plot the horizontal vectors in the x, y coordinates.

getxxyy Fills the **xs**, **ys**, **xv**, **yv**, **xb**, and **yb** arrays for **fx**, **fy**, and **cpmpxy**.

label Puts labels on plots. Changes the **set** plotting environment.

mxf, myf Used by **velvct** to plot vectors in x, y coordinates.

peplt Creates slab plots of $u, v, \psi, \rho, T, S, w, \zeta, \vec{v}, \phi$ and calls **pltslca** and **pltslcb** for the slice plots.

ploth Plots bottom topography, its gradient, and a resolution ratio (see Appendix C).

pltperim Draws the outline of the model domain in x, y coordinates. Used with **velvct**.

pltslca Plot slices on surfaces of constant ξ.

pltslcb Plot slices on surfaces of constant η.

velvct A modified version of the NCAR Graphics velocity vector subroutine, allowing velocity vectors to be plotted in x, y space instead of ξ, η space. See section 4.5 and Appendix D.

xyzplot Fills an array for plotting horizontal sections of the model fields on a level of constant z.

yzplot Fills an array for plotting vertical sections of the model fields in y, z or x, z space.

Lagrangian floats

bcw Boundary conditions on w (see section 6.1.1).

fltinit Initializes the particle positions, called from the main program after **init**. The particle positions should be specified to lie in the range $1 \leq \xi \leq \mathbf{L}$, $1 \leq \eta \leq \mathbf{M}$, and $-1 \leq \sigma \leq 1$.

fltstp Called by the main program to advance the particles.

rhoflt Subroutine which returns ρ at the float positions (ξ, η, σ) by using the equation of state $\rho(T, S, z)$.

rk4 Fourth order Runge-Kutta implementation derived from *Numerical Recipes* by Press, et al. [12]. Calls **xderiv**, **yderiv**, and **zderiv** or **x2deriv**, **y2deriv**, and **z2deriv**.

sflt Subroutine which returns S at the float positions (ξ, η, σ) by a bicubic interpolation of S in a 4×4 box.

s2flt Subroutine which returns S at the float positions (ξ, η, σ) by a bilinear interpolation of S in a 2×2 box.

tflt Subroutine which returns T at the float positions (ξ, η, σ) by a bicubic interpolation of T in a 4×4 box.

t2flt Subroutine which returns T at the float positions (ξ, η, σ) by a bilinear interpolation of T in a 2×2 box.

xderiv Subroutine which returns $d\xi/dt$ at the float positions (ξ, η, σ) by a bicubic interpolation of $u \cdot m$ in a 4×4 box.

x2deriv Subroutine which returns $d\xi/dt$ at the positions (ξ, η, σ) by a bilinear interpolation of $u \cdot m$ in a 2×2 box.

yderiv Subroutine which returns $d\eta/dt$ at the positions (ξ, η, σ) by a bicubic interpolation of $v \cdot n$ in a 4×4 box.

y2deriv Subroutine which returns $d\eta/dt$ at the positions (ξ, η, σ) by a bilinear interpolation of $v \cdot n$ in a 2×2 box.

zderiv Subroutine which returns $d\sigma/dt$ at the float positions (ξ, η, σ) by a bicubic interpolation of Ω in a 4×4 box.

z2deriv Subroutine which returns $d\sigma/dt$ at the positions (ξ, η, σ) by a bilinear interpolation of Ω in a 2×2 box.

Other

vmax Returns the largest element in a vector.

vmin Returns the smallest element in a vector.

4.3 Important parameters

Following is a list of the important parameters in the model. Most other parameters are derived from **L**, **M**, and **N**.

L Number of grid points in the ξ direction.

M Number of grid points in the η direction.

N Number of baroclinic polynomials in the vertical.

nflt Number of Lagrangian floats.

nwrk Size of the work space used by **mud2**.

4.4 Common blocks and the variables within them

All variables which have units are given in MKS units.

BLANK The main time-dependent model fields

time	time
u	velocity component in ξ direction
v	velocity component in η direction
w	Ω, velocity component in σ direction
rho	perturbation density (total density = $\rho_o + \rho$)
t	temperature T
s	salinity S
psi	horizontal transport streamfunction ψ
vrt	vorticity of depth-averaged flow field ζ
dwds	vertical derivative of Ω, really $\frac{\partial}{\partial\sigma}\left(\frac{h\Omega}{mn}\right)$ or $\frac{h}{mn}\frac{\partial\Omega}{\partial\sigma}$

/chb/ Polynomial transformation matrices

pi	$\pi = 3.14159265\ldots$
sig	array of sigma values $\sigma(k)$
cp	matrix of orthogonal polynomials $F = P_k(\sigma_n)$ (equation 3.3). Multiply a mode amplitude vector by **cp** to get the vector of values at the σ_n locations, $b(\sigma_n)$.
cf	inverse of the matrix of orthogonal polynomials, F^{-1} (equation 3.5). Multiply a $b(\sigma_n)$ vector by **cf** to get the mode amplitude vector.
cd	derivatives of $P_j(\sigma_i)$ (equation 3.6)
cdz	derivative transform matrix C_{DZ} (equation 3.7). Multiply a vertical vector by **cdz** to get the vector of derivatives.
cint	integral transform matrix C_{INT} (equation 3.9). Multiply a vertical vector by **cint** to get the vector of integrals, integrating down from the surface.
csum	integrals of $P_k(\sigma)$ over $-1 \leq \sigma \leq +1$

/climat/ Model climatology used in nudging terms

tclm	T climatology
sclm	S climatology
uclm	u climatology
vclm	v climatology

/coord/ Orthogonal coordinate transformation arrays

pm coordinate transformation metric m

pn coordinate transformation metric n

dndx $\frac{\partial}{\partial \xi}(\frac{1}{n})$

dmde $\frac{\partial}{\partial \eta}(\frac{1}{m})$

/ellip/ Parameters needed for elliptic solver. See the **mud2** documentation for details

iprm parameters for the **mud2** elliptic solver

iprm parameters for the **mud2** elliptic solver

ewrk work array of size **nwrk**

nmud number of steps between printing out the **mud2** error

mgopt parameters for the **mud2** elliptic solver

/flags/ Logical arrays for the right hand side calculations

uflags logical array used to determine which terms are to be included in the u equation

vflags logical array used to determine which terms are to be included in the v equation

sflags logical array used to determine which terms are to be included in the S equation

tflags logical array used to determine which terms are to be included in the T equation

mixflag logical flag to turn on vertical mixing

stpflag logical flag to determine the type of time step corrections to take. **true** is for third-order Adams-Bashforth and **false** is for leapfrog-trapezoidal steps.

eqstflag logical flag to determine which equation of state to use. **true** for a non-linear $\rho(T, S, z)$ and false for a linear $\rho(T)$

/floats/ Lagrangian float variables

flt A two-dimensional array containing the particle information. The first index specifies which particle is being referred to and has a range of 1 to **nflt**. The second index specifies the field according to the key:

$$
\begin{aligned}
1 &= \xi \\
2 &= \eta \\
3 &= \sigma \\
4 &= \tfrac{d\xi}{dt}(\xi, \eta, \sigma) = u \cdot m \\
5 &= \tfrac{d\eta}{dt}(\xi, \eta, \sigma) = v \cdot n \\
6 &= \Omega(\xi, \eta, \sigma) \\
7 &= T(\xi, \eta, \sigma) \\
8 &= S(\xi, \eta, \sigma) \\
9 &= \rho(\xi, \eta, \sigma)
\end{aligned}
$$

kptime Interval between disk writes of float information

fltflag Logical flag to specify whether or not floats are being used

flt4flag Logical flag to specify 4×4 or 2×2 horizontal interpolation. **true** for 4×4.

nfltrrec Number of float restart records to read in from disk during initialization

/grdpts/ x, y (physical space) gridpoint locations, used in plotting the model fields

xp x coordinates of the ψ points

yp y coordinates of the ψ points

xmin minimum x value on the ψ grid

ymin minimum y value on the ψ grid

xmax maximum x value on the ψ grid

ymax maximum y value on the ψ grid

xl length of domain in ξ direction, **xl = xmax − xmin**

el length of domain in η direction, **el = ymax − ymin**

/id/ Descriptive identification for the current simulation

ident 160 character string, first 24 characters are used to label plots

/ionum/ Unit numbers for I/O. The SPEM does not specifically open files or use any particular file names. Fortran compilers have default file names for the unit numbers, such as **fort.3** or **FOR001.DAT** and these are the file names used by SPEM for its binary files.

iin unit number for reading in history file

iout unit number for writing out history file

/parm/ Physical constants and fields

h bottom depth

f Coriolis parameter $= 2 \cdot \Omega_{\text{Earth}} \sin(\theta)$

uvnu2 lateral Laplacian mixing coefficient (constant) for u, v

tnu2 lateral Laplacian mixing coefficient (constant) for T

snu2 lateral Laplacian mixing coefficient (constant) for S

uvnu4 lateral biharmonic mixing coefficient (constant) for u, v

tnu4 lateral biharmonic mixing coefficient (constant) for T

snu4 lateral biharmonic mixing coefficient (constant) for S

rho0 mean density ρ_o

g acceleration due to gravity

rdrg bottom drag coefficient

hmin minimum depth of the topography

hmax maximum depth of the topography

dhdx $\frac{1}{n}\frac{\partial h}{\partial \xi}$

dhde $\frac{1}{m}\frac{\partial h}{\partial \eta}$

rfac $\frac{g}{2\rho_o}$

rdmp inverse timescale of the nudging towards climatology

/**perbcs**/ Variables particular to periodic boundary conditions

perchan logical flag—**true** for periodic boundary conditions in the x direction

perintg logical flag used only in the periodic channel version—**false** if you will specify ψ on both channel walls, **true** if you want the model to calculate the channel transport

gfpy used to calculate along-channel momentum balance

py used to calculate along-channel momentum balance

grnfnc the transport Green's function for a periodic channel

/**pg**/ Pressure gradient arrays and parameters

phix ξ component of the baroclinic pressure gradient

phie η component of the baroclinic pressure gradient

sphix ξ component of the surface pressure gradient

sphie η component of the surface pressure gradient

npgc number of pressure gradient correction terms to calculate

nspr number of steps between calculating the surface pressure gradients

/**plt**/ Plotting flags and parameters

nplvl number of levels to be plotted

npl array of specific σ levels to be plotted, from 0 to **N**

nslcea number of y-z cross sections to be plotted

nsla array of specific ξ-locations of the y-z cross sections

nslceb number of x-z cross sections to be plotted

nslb array of specific η-locations of the x-z cross sections

plot u logical switch to turn on plotting of the u field

plot v logical switch to turn on plotting of the v field

plot s logical switch to turn on plotting of the S field

plot t logical switch to turn on plotting of the T field

plot w logical switch to turn on plotting of the w field

plot rho logical switch to turn on plotting of the ρ field

plot psi logical switch to turn on plotting of the ψ field

plot vort logical switch to turn on plotting of the ζ field

plot vec logical switch to turn on plotting of the $\vec{v}$ field

plot spr logical switch to turn on plotting of the surface pressure gradient

zslab determines whether the level (slab) plots are of planes of constant z or constant σ. **true** for plots at a constant z

zplot array of specific z levels to be plotted

ztop the z-level of the top of the slice plots

zbot the z-level of the bottom of the slice plots

rbarsub logical flag to specify whether **rhobar** is subtracted out before ρ is plotted

/stp/ Time step parameters (see Appendix A)

dt time step Δt

ntmes number of steps in current run

nrst number of steps between storage of restart fields

nrrec number of restart records previously written to disk

nplt number of steps between plots (calls to subroutine **peplt**)

nmix number of steps between vertical mixings

ncorr number of steps between correction steps

iic time step counter

jjc time step component counter

lnew one of several time level indices (equal to either 1 or 2) used for the last array index in the model fields. **lnew** points to the current time level

lold pointer to the previous time level

lstp pointer to the time level to which the current changes are added

ldis pointer to the time level used in the dissipative terms (not **lrhs** for numerical stability reasons)

lrhs pointer to the time level used to calculate the right-hand side terms

ltrp pointer to the time level of the right hand side arrays (**ru**, **rv**, etc.) to which the changes are being added

lab1 pointer to a time level for the right hand side arrays

lab2 pointer to a time level for the right hand side arrays

cfact0 weighting factor for the right hand side arrays

cfact1 weighting factor for the right hand side arrays

cnb weighting factor for $\mathcal{F}(t - \Delta t)$

cnc weighting factor for $\mathcal{F}^*(t + \Delta t)$

dts size of the timestep between **lstp** and **lnew**

/user/ Arrays, variables, and constants defined by the user as being relevant to a specific application

/vec2/ velvct internal common block — allows the plotting of a velocity vector every nth grid point

incx plots every **incx** vector in the x direction

incy plots every **incy** vector in the y direction

/vrtadv/ Needed for energy conservation operations

nrgcons logical flag—**true** if you want to conserve energy

c3d 3-d work array

/wrk/ Temporary work/storage arrays

rs stores right hand side of S equation

rt stores right hand side of T equation

ru stores right hand side of u equation

rv stores right hand side of v equation

rz stores right hand side of vorticity equation

tmp 2-d work array. **tmp** and **tmp2** are used to carry the depth-averaged forcing from **ustp** and **vstp** to **vorteq** and should not be changed in that part of the main loop

tmp2 2-d work array

wrz 1-d (vertical) work array

wvz 1-d (vertical) work array

a3d 3-d work array

b3d 3-d work array

/xxyys/ Temporary x, y arrays for the plots

xs x position for **fx** in the vertical y, z slice plots

ys y position for **fy** in the vertical y, z slice plots

xv x position for **fx** in the horizontal slab plots

yv y position for **fy** in the horizontal slab plots

xb x position for **fx** in the vertical x, z slice plots

yb y position for **fy** in the vertical x, z slice plots

Other variables

akuv vertical diffusive coefficient for u, v (statement function of i, j, k)

akt vertical diffusive coefficient for T (statement function of i, j, k)

aks vertical diffusive coefficient for S (statement function of i, j, k)

lspace determines whether the slab plots are of constant σ levels or the mode amplitude. **lspace** = **true** will produce plots in physical space.

4.5 Changes to velvct

A number of changes have to be made to the NCAR graphics [6] routine **velvct** in order for the vector plots to work in the SPEM. Appendix D shows the UNIX 'context differences' between the unmodified and the modified versions of **velvct.**

The most important change allows the plots to be made in the curvilinear coordinates instead of rectangular coordinates. This is done by commenting out the lines

```
FX(X,Y) = X
FY(X,Y) = Y
```

and the lines

```
MXF(XX,YY,UU,VV,SFXX,SFYY,MXX,MYY) = MXX + IFIX(SFXX*UU)
MYF(XX,YY,UU,VV,SFXX,SFYY,MXX,MYY) = MYY + IFIX(SFYY*VV)
```

and providing the functions **fx**, **fy**, **mxf**, and **myf**.

Another change to **velvct** allows the vectors to go outside the model boundary. Add the lines

```
call gqclip(IER,ICLP,IAR)
call gsclip(0)
```

before the line

```
C DRAW THE VECTORS
```

This will save the current clipping state and disable the GKS clipping. The two lines

```
CALL GQCLIP(IER,ICLP,IAR)
CALL GSCLIP(0)
```

a few lines down should be commented out so that the saved clipping state is not overwritten.

Chapter 5

Configuring SPEM for a Specific Application

This chapter describes the parts of SPEM for which the user is responsible when configuring it for a given application. Section 5.1 describes the process in a generic fashion while section 5.2 steps through the application of SPEM to a seamount resonance problem. As distributed, SPEM is ready to run the seamount example in a periodic channel. Section 5.3 describes the changes which are needed for a non-periodic application.

5.1 Configuring SPEM

5.1.1 Model domain

One of the first things the user must decide is how many grid points to use (and can be afforded). There are three parameters which specify the grid size:

- **L** Number of finite-difference points in ξ.
- **M** Number of finite-difference points in η.
- **N** Number of baroclinic vertical modes.

Since the model uses the **mud2** elliptic solver you are not free to pick just any values for **L** and **M**. For a non-periodic domain the following relations must be satisfied:

$$\mathbf{L} = \mathbf{nxl} \times 2^{(\mathbf{nstep}-1)} + 1 \tag{5.1}$$

$$\mathbf{M} = \mathbf{nyl} \times 2^{(\mathbf{nstep}-1)} + 1 \tag{5.2}$$

where the solver is more efficient for higher values of **nstep**, since **nstep** defines the number of sub-grid levels used in the multi-grid algorithm. If periodic boundary conditions are being used then the **L** relation becomes:

$$\mathbf{L} = \mathbf{nxl} \times 2^{(\mathbf{nstep}-1)} + 2. \tag{5.3}$$

See the **mud2** documentation for a description of the **mud2** solver.

L, **M**, and **N** are parameters which are used in almost every subroutine. They should be set to your chosen values throughout the code. This is also a good time to initialize the appropriate parameters for **mud2**. In **peminit** set

$$\mathbf{iprm}(6) = \mathbf{nxl}$$
$$\mathbf{iprm}(7) = \mathbf{nyl}$$
$$\mathbf{iprm}(8) = \mathbf{nstep}.$$

The size of the work array **ewrk** is given by the parameter **nwrk**. For periodic domains **nwrk** should be set to $16 \cdot 4 \cdot \mathbf{L} \cdot \mathbf{M}/3$ while for open domains the size $14 \cdot 4 \cdot \mathbf{L} \cdot \mathbf{M}/3$ is sufficient.

5.1.2 x, y grid

The subroutine **getgrid** is called by **peminit** to read in the grid arrays, the topography, and the Coriolis parameter. The file it reads is produced by the grid generation programs described in Hedstrom and Wilkin [10]. The variables which are read by the subroutine **getgrid** are:

pm, pn, dndx, dmde, xp, yp, f, h.

From these fields **getgrid** calculates the related variables:

xmin, xmax, ymin, ymax, xl, el, hmin, hmax, dhdx, dhde.

The aspect ratio of the domain is given by **el/xl**. The plotting subroutines currently assume that the aspect ratio will be less than 2.5. If this is not the case then you can adjust the variables **x1**, **x2**, **y1** and **y2** in the subroutines **peplt** and **ploth** for your domain.

For a simple rectangular geometry a short program called **ezgrid** can be used to calculate the required grid arrays. **ezgrid** is used in the seamount example described in section 5.2.

5.1.3 ξ, η grid

Before providing initial conditions and boundary conditions the user must understand the model grid. The fields are laid out on an Arakawa C grid as in figure 3.1. The overall grid is shown in figure 5.1. The thick outer line shows the position of the model boundary. The points inside this boundary are those which are advanced in time using the model physics. The points on the boundary and those on the outside must be supplied by the boundary conditions.

The three-dimensional model fields are carried in four-dimensional arrays where the fourth array index refers to the two time levels (see appendix A). The integers **i**, **j**, and **k** are used throughout the model to index the three spatial dimensions:

- **i** Index variable for the ξ direction.
- **j** Index variable for the η direction.
- **k** Index variable for the σ direction. $\mathbf{k} = 0$ refers to the bottom while $\mathbf{k} = \mathbf{N}$ refers to the surface.

The range of ξ is 1 to **L** and the range of η is 1 to **M**. Therefore **i** and ξ are the same at ψ points, as are **j** and η.

5.1.4 Initial conditions

The initial values for the model fields are provided by **peminit**. Only the interior points of u, v, T, S and ψ need be initialized in **peminit** because **bcs** will be called after **peminit**.

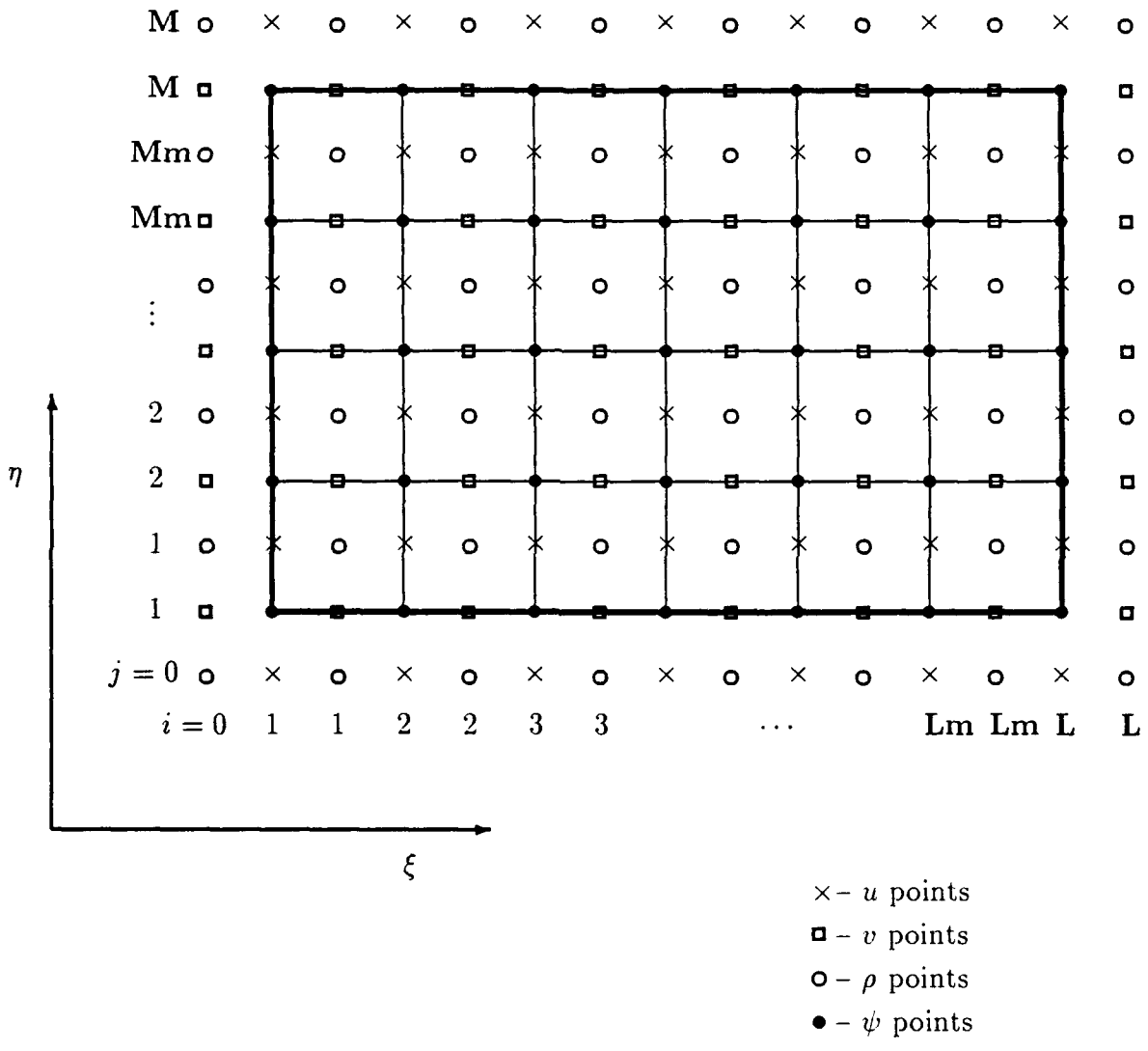


Figure 5.1: The whole grid

5.1.5 Equation of state

The equation of state is used in the subroutines **rhocal** and **rhoflt**. Two versions are provided in SPEM as it is distributed: a nonlinear $\rho = \rho(T, S, z)$ and a linear $\rho = \rho(T)$. It is possible to provide your own equation of state, but if you use floats you should make sure that **rhocal** and **rhoflt** are consistent. Also, it is a good idea to write it in a vectorizable form if you run SPEM on a vector machine.

5.1.6 rhobar and rhobarz

The statement function **rhobar** is used in **prsgrd** to designate a density stratification which is a function of z only and which will be subtracted out before the calculation of the pressure gradients. **rhobar** is also set in some of the plotting routines to allow the plotting of $\rho - \overline{\rho}$ rather than the plotting of ρ alone. There is a related statement function **rhobarz** where $\overline{\rho}$ is expressed as a function of z as opposed to **i**, **j**, and **k**.

5.1.7 Climatology

The climatology fields **uclm**, **vclm**, **tclm** and **sclm** contain the values to whicl. u, v, T and S are nudged when the body forcing is turned on. These arrays are initialized in **peminit**. The inverse nudging timescale **rdmp** is also initialized in **peminit**.

5.1.8 Boundary conditions

The horizontal boundary conditions are provided by the subroutine **bcs** and its entry **bcpsi**. They are called for every timestep, including during the initialization. and must provide the boundary values for the fields u, v, T, S and ψ. The user configures this subroutine to provide the boundary conditions desired.

In the special case of the periodic version (**perchan = true**) the subroutine **bcs** sets the boundary values to wrap around appropriately. Values at **i = L** are set to those at **i = 2**, values at **i = 0** are set to those at **i = Lm2**, and values at **i = 1** are set to those at **i = Lm**. There is an extra set of ρ points and v points for compatibility with the open version.

The elliptic solver also has to know what type of boundary conditions to provide. These are specified by **iprm(2)** through **iprm(5)** and are initialized in **peminit**.

5.1.9 Timestep

The variable **dt** is initialized in **peminit**. For simple wave-like problems the overall stability is limited by a CFL-type condition such that $c\frac{\Delta t}{\Delta x} < 1$. However, when horizontal and vertical diffusion are included, the stability issue becomes more complicated.

It is sometimes useful to be able to change the timestep and/or the horizontal friction parameters in the middle of a model run. When the model is restarted from a history file, values for **dt**, **uvnu4**, etc. are read in. These old values will have to be overwritten in **init** after the call to **histio**.

5.1.10 Model Input

ident A string 160 characters long used to identify the experiment which is read in on unit 5. The first 24 characters are used to label the plots.

5.1.11 Block data

As many of the user choices as possible are put into the block data at the end of the program. They are as follows and are described more fully in section 4.4.

sflags Logical flags to tell the model which terms in equation 2.22 to use. See the note in the block data for details.

tflags Logical flags to tell the model which terms in equation 2.21 to use.

uflags Logical flags to tell the model which terms in equation 2.19 to use.

vflags Logical flags to tell the model which terms in equation 2.20 to use.

ntmes Number of time steps to take.

stpflag Logical flag to specify the type of the correction timestep.

ncorr How often to take a correction step.

nrrec Number of restart records to read in from disk.

mixflag *Logical flag which is true if vertical mixing of statically unstable fields is desired.*

nmix How often to mix vertically.

eqstflag Logical flag which is true if the non-linear equation of state is desired.

nplt, nrst, nmud How often to produce various types of output.

g, rho0 Environmental parameters.

rdrg, uvnu2, snu2, tnu2, uvnu4, snu4, tnu4 Frictional parameters.

perchan, perintg Logical flags for the periodic channel boundary conditions.

nspr How often to calculate surface pressure gradients.

npgc Number of corrections to the pressure gradient terms to compute.

nrgcons Logical flag for quadratic moment conservation.

fltflag, flt4flag, nfltrrec, kptime Lagrangian float information.

nplvl, zslab, npl, zplot Control of horizontal level plots.

nslcea, nsla, nslceb, nslb, zbot, ztop Control of vertical slice plots.

rbarsub Logical flag which is **true** if $\rho - \overline{\rho}$ should be plotted instead of ρ.

plot ... Control which model fields are plotted.

5.1.12 Vertical diffusion

The vertical diffusive coefficients **akt**, **aks**, and **akuv** can vary over space. They are currently implemented as statement functions within **trhs**, **srhs**, **urhs**, and **vrhs**. If these coefficients are sufficiently complicated and if enough memory is available, it is sometimes more efficient to implement them as three-dimensional arrays, as described in section 5.1.13.

5.1.13 User variables

It is possible for the user to add new variables and common blocks appropriate to a given application, for instance, some boundary conditions require that you keep track of more information than the model currently stores. These new variables can be initialized in **peminit** or **init**, but recall that **peminit** is not called when restarting from a history file. Variables which are initialized in **peminit** should be written out to the history file, either in **histio** (if they do not change), or in **oput**.

5.2 Seamount Resonance Example

The particular application for which we have configured the model includes a diurnal tide over a tall seamount in a stratified ocean. The question is whether or not the tide will excite a resonance with seamount-trapped waves which have a period close to one day. If so, this could explain the large bottom-trapped currents observed over Fieberling Guyot.

We have chosen to place the seamount in a periodic channel to simplify the boundary conditions. The grid is rectangular with the finest resolution over the seamount and coarser resolution elsewhere. The tidal flow is provided by the boundary conditions, specifically the value of streamfunction at the far wall.

5.2.1 Model domain

A relatively small number of grid points were chosen for reasons of economy. Values for **L**, **M**, and **N** are:

$$\begin{aligned} \mathbf{L} &= 42 \\ \mathbf{M} &= 41 \\ \mathbf{N} &= 7. \end{aligned}$$

These values lead to the **mud2** parameters of

$$\begin{aligned} \mathbf{iprm}(6) &= \mathbf{nxl} = 5 \\ \mathbf{iprm}(7) &= \mathbf{nyl} = 5 \\ \mathbf{iprm}(8) &= \mathbf{nstep} = 4. \end{aligned}$$

5.2.2 ezgrid

For this geometry one has a choice of using the grid-generation programs described in Hedstrom and Wilkin [10] or of creating a stand-alone program to create the grid file analytically.

A short program called **ezgrid** was written to produce a uniform rectangular grid file. This program was modified to produce a stretched grid with a Gaussian seamount when the variable **stretch** is **true**. The fluid depth is 5000 meters and the seamount is 4500 meters tall.

To use **ezgrid** for another application, read the comments in the code and make the appropriate changes. For a grid with constant grid spacing, set **stretch** to false and supply the size of the domain in the parameters **xl** and **el**. You will also need to provide it with **h(i,j)** and **f(i,j)**.

5.2.3 Initial conditions and the equation of state

We would like the initial conditions to be motionless fluid with an exponential stratification. The statement functions **ufld**, **vfld**, and **psifld** are all set to zero.

The stratification can be provided by T, S, or both. For simplicity we will only have an active temperature field and we will use the linear equation of state where density is a function of temperature only. We want the density to be 28.0 on the bottom and 27.5 at the top with an e-folding scale of 1000 meters. The initial temperature is set to $1.43 + 3.57e^{z/1000}$ in **tfld**.

Since density does not depend on salinity, we have a choice on how to handle the salinity field. We can either use it as a passive tracer or not timestep on it at all by setting **sflags(7)** to **false**. We will use it as a passive tracer and initialize it in **sfld** to be a function of y.

5.2.4 rhobar and rhobarz

We have set **rhobar** and **rhobarz** to the density field derived from the initial temperature field.

5.2.5 Climatology

We have turned nudging off for this calculation so the climatology fields are not used.

5.2.6 Boundary conditions

The boundary conditions are walls to the north and south ($\eta = 1.\mathrm{M}$). The walls can be implemented simply by specifying no flow out of the box:

$$v = 0 \quad @ \quad j = 1.\mathrm{M}.$$

In the presence of horizontal diffusion it is also necessary to provide horizontal momentum and density fluxes through the walls. These fluxes are determined by the values of u and ρ outside the boundary. No flux boundaries were chosen by setting $u_{\text{outside}} = u_{\text{inside}}$ and $\rho_{\text{outside}} = \rho_{\text{inside}}$.

The boundaries to the east and west are periodic. **bcs** is configured to provide periodic boundary conditions when **perchan** is **true**.

Finally, the boundary conditions provide the barotropic tidal flow. The total transport through the domain is given by the difference between the values of ψ on the northern and

southern walls. We can tell SPEM that we will be providing the value of ψ on the northern wall by setting **perintg** to **false** (the southern wall is always set to zero).

We want the velocity far from the seamount to be

$$u = U_o \sin(\omega t)$$

where U_o is 1 mm/sec and ω is $2\pi/1$day. The corresponding value of Ψ_o can be gotten by multip'ying U_o by the cross-sectional area of the channel (5000 m by 320 km). This leads to $\Psi_o = 1.6 \times 10^6$ or 1.6 Sverdrups. We do not want to shock the system by turning on the forcing all at once so we ramp it up with a

$$\frac{1}{2}\left[1 + \tanh\left(\frac{t - t_o}{\alpha}\right)\right]$$

over a period of several days.

5.2.7 Timestep

The timestep **dt** is initialized in **peminit** to be a day over 128.

5.2.8 Block data

The terms to be used in the equations of motion are specified by the flags in the block data. We want the Coriolis, pressure gradient, advection and biharmonic friction in the momentum equations so **uflags** and **vflags** 2, 5, 7 and 8 are **true** while the rest are **false**. The value of the biharmonic friction coefficient **uvnu4** is set to 1×10^{10}. We want an adiabatic calculation so only the temperature advection **tflags(5)** is **true**. To enable salinity timestepping and advection we set **sflags** 5 and 7 to **true**.

The variable **perchan** is **true** for periodic boundary conditions in ξ, but set **perintg** to **true** because the **bcs** is supplying the channel transport.

eqstflag is **false** to get a linear equation of state.

The model has been set up to run for ten days or 1280 timesteps.

5.2.9 Output

The model writes some information to standard out:

```
read grid file
Stretched grid for the seamount example
  Seamount Example                Run on Cray 2
May 15, 1990

L    =  42
M    =  41
N    =   7
xl   =    3.311E+05
el   =    3.201E+05
```

```
rho0 =    1.000E+03
g    =    9.810E+00
rdrg =    3.000E-04
hmin =    5.000E+02
hmax =    5.000E+03
dt   =    6.750E+02
uvnu2=    1.000E+03
snu2 =    1.000E+03
tnu2 =    1.000E+03
uvnu4=    1.000E+10
snu4 =    1.000E+10
tnu4 =    1.000E+10
rdmp =    0.000E+00
initial error for greens function solution =    8.618E-05

init call to mud2: ierr = -1


Plotting for day:        0.00  has been done

    64   3.426E-03
   128   3.072E-03
   192   2.719E-03
   256   2.402E-03
   320   2.289E-03
   320   3.962E-06
   384   2.203E-03
   448   2.131E-03
   512   2.068E-03
   576   2.004E-03
   640   1.936E-03
   640   3.344E-06
   704   1.862E-03
   768   1.779E-03
   832   1.694E-03
   896   1.598E-03
   960   1.505E-03
   960   2.591E-06
  1024   1.410E-03
  1088   1.329E-03
  1152   1.246E-03
  1216   1.183E-03
  1280   1.119E-03
  1280   1.913E-06
```

```
Plotting for day:        10.00  has been done

    2 restart records written
```

A binary restart/history file is also produced. It contains the model fields at times 0 and 10 days as well as all the other information needed by the model to continue the run where it left off.

The final output is an NCAR graphics metafile. The pictures from this file are shown in figures 5.2 to 5.18.

5.3 Changes needed for an open domain

5.3.1 mud2

First of all, the number of grid points is different for a periodic domain, as described in section 5.1.1. The **L** dimension must be one less to be consistent with equation (5.1). The size of **nwrk** can also be made smaller. For a periodic channel

$$\mathbf{nwrk} = 16 \cdot 4 \cdot \mathbf{L} \cdot \mathbf{M}/3$$

while for an open domain

$$\mathbf{nwrk} = 14 \cdot 4 \cdot \mathbf{L} \cdot \mathbf{M}/3$$

is sufficient. See the **mud2** *documentation.*

5.3.2 Block data

The variable **perchan** should be set to **false** in the blockdata.

5.3.3 Boundary conditions

The subroutine **bcs** must be changed to provide the desired boundary conditions. The default is a closed basin when **perchan** is **false**.

5.3.4 ezgrid

In **ezgrid** the line

```
dx = xl / float(Lm2)
```

should be changed to

```
dx = xl / float(Lm).
```

In **ezgrid** the variable **xl** is the length of the periodic domain which goes from $\xi = 1$ to $\xi =$ **Lm**. In the SPEM the variable **xl** is the length of the channel from $\xi = 1$ to $\xi =$ **L** and is only used to scale the plots.

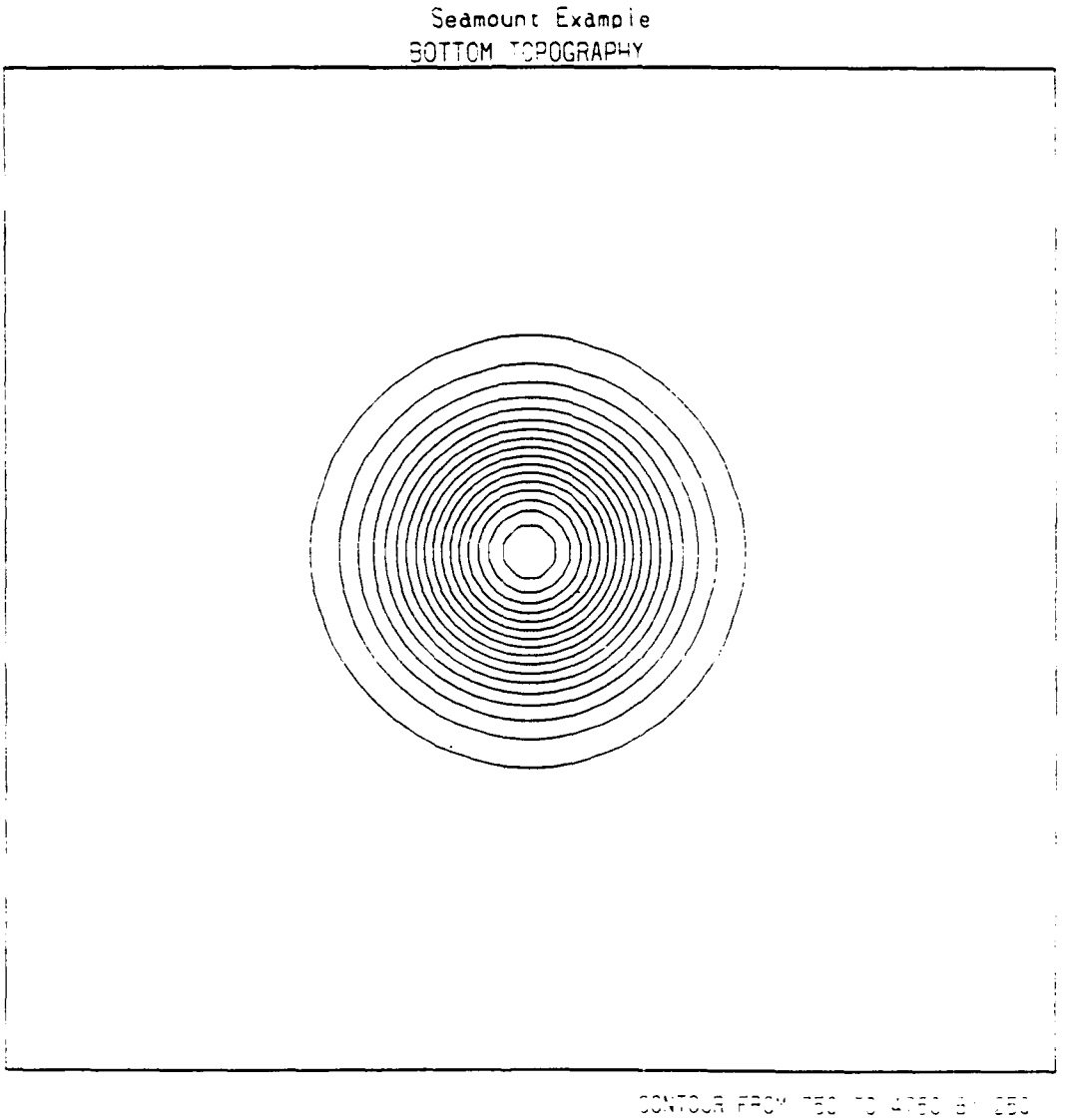


Figure 5.2: Bottom topography

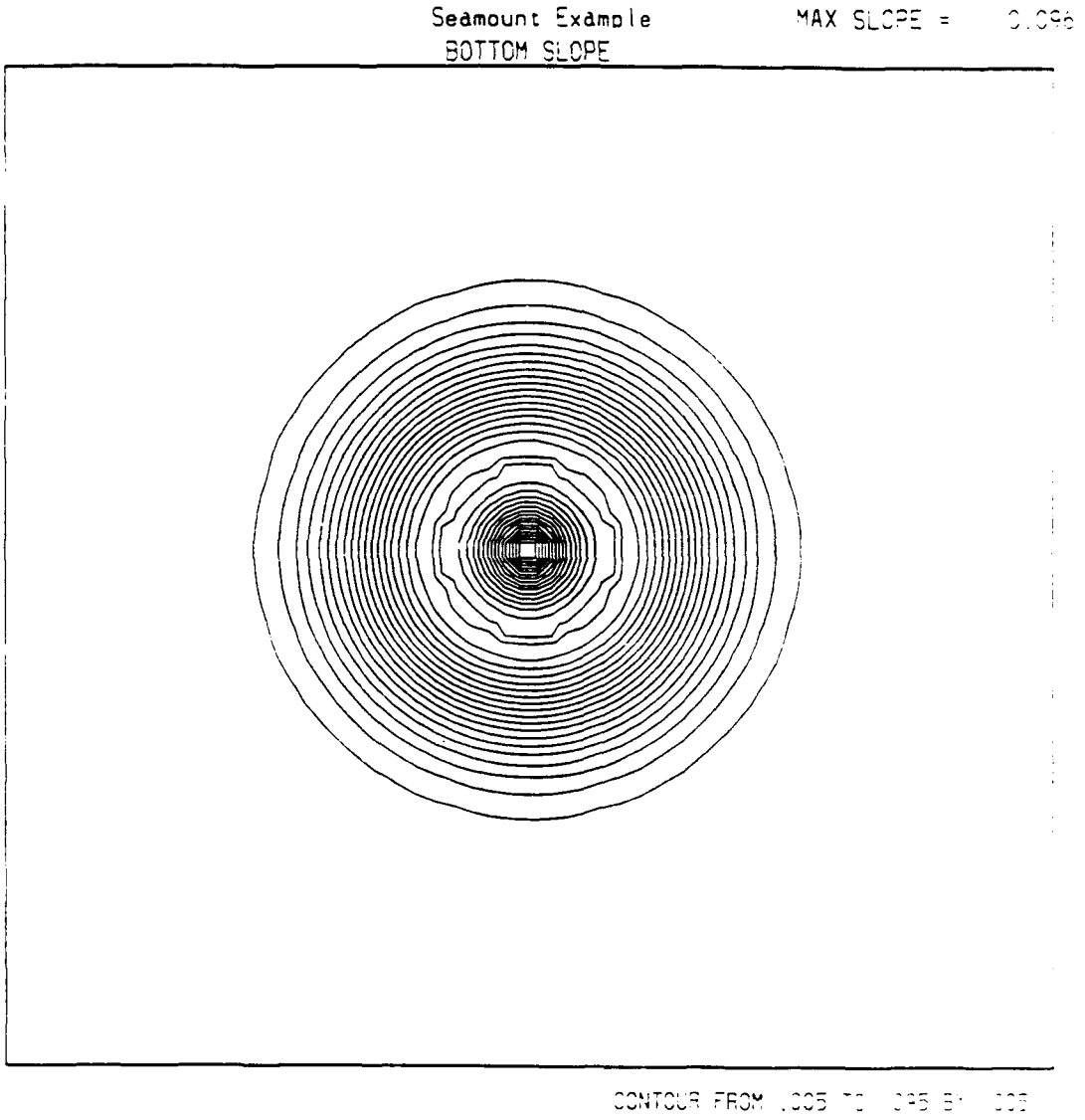


Figure 5.3: Gradient of topography

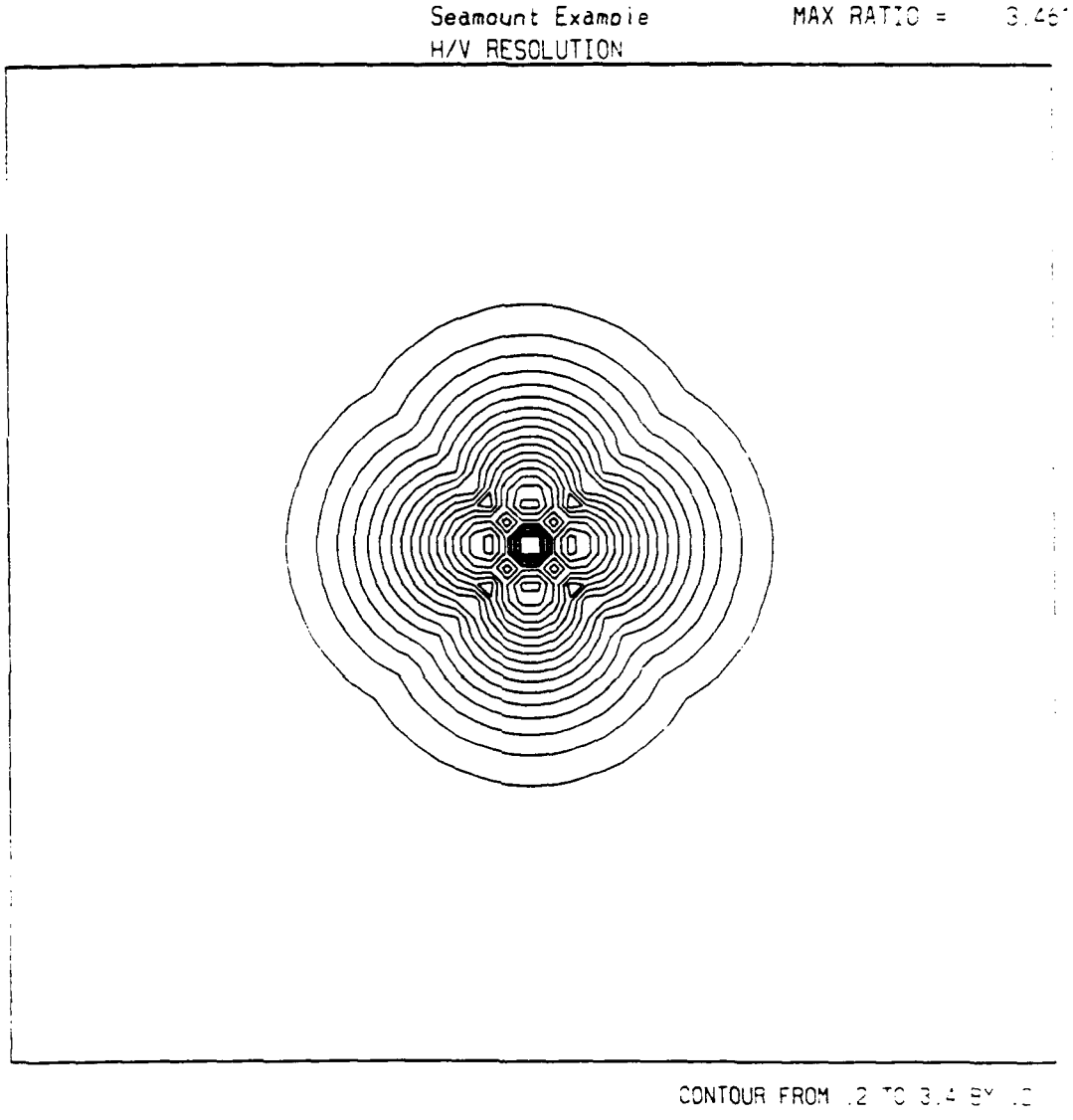


Figure 5.4: Resolution factor

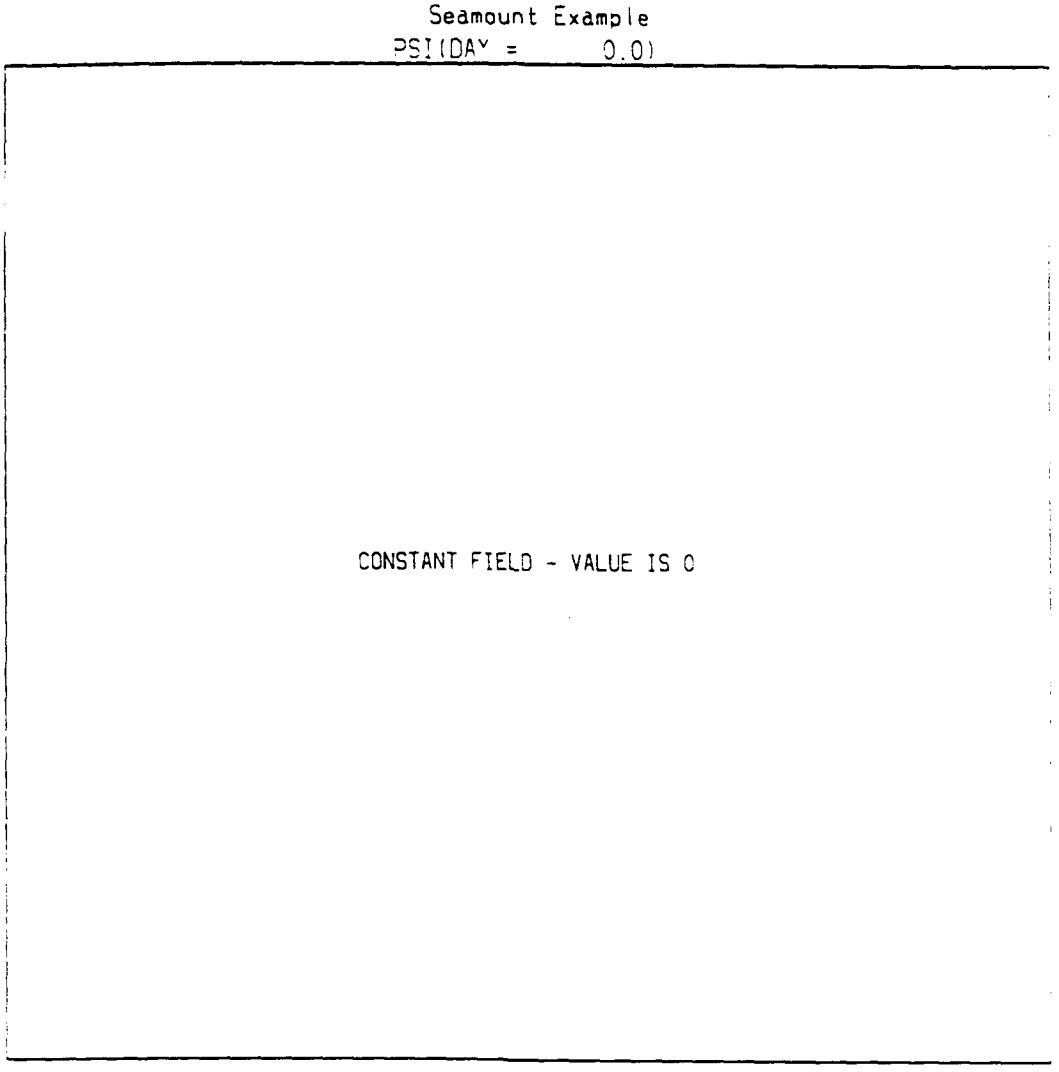


Figure 5.5: ψ field at day 0

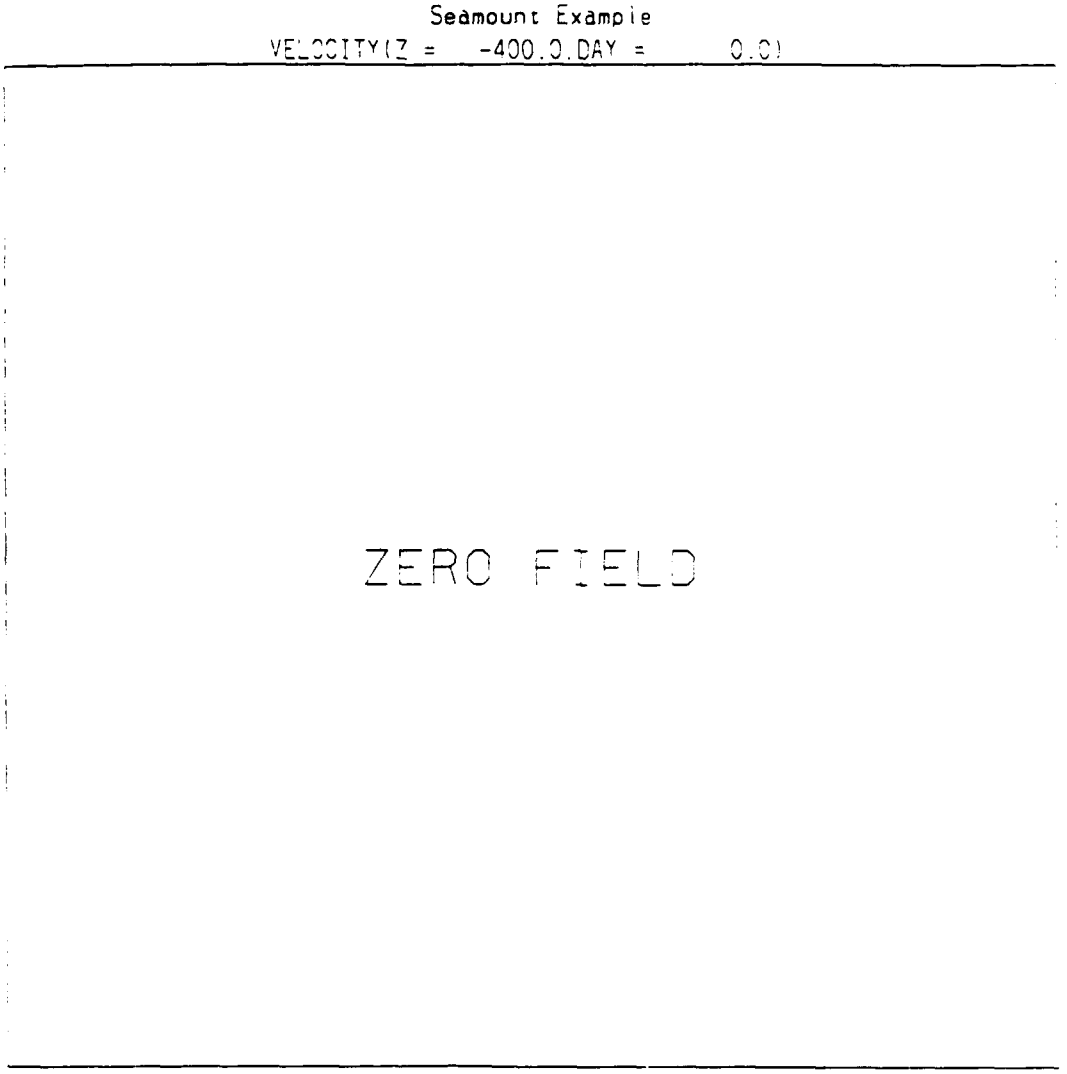


Figure 5.6: $\vec{v}$ field at day 0

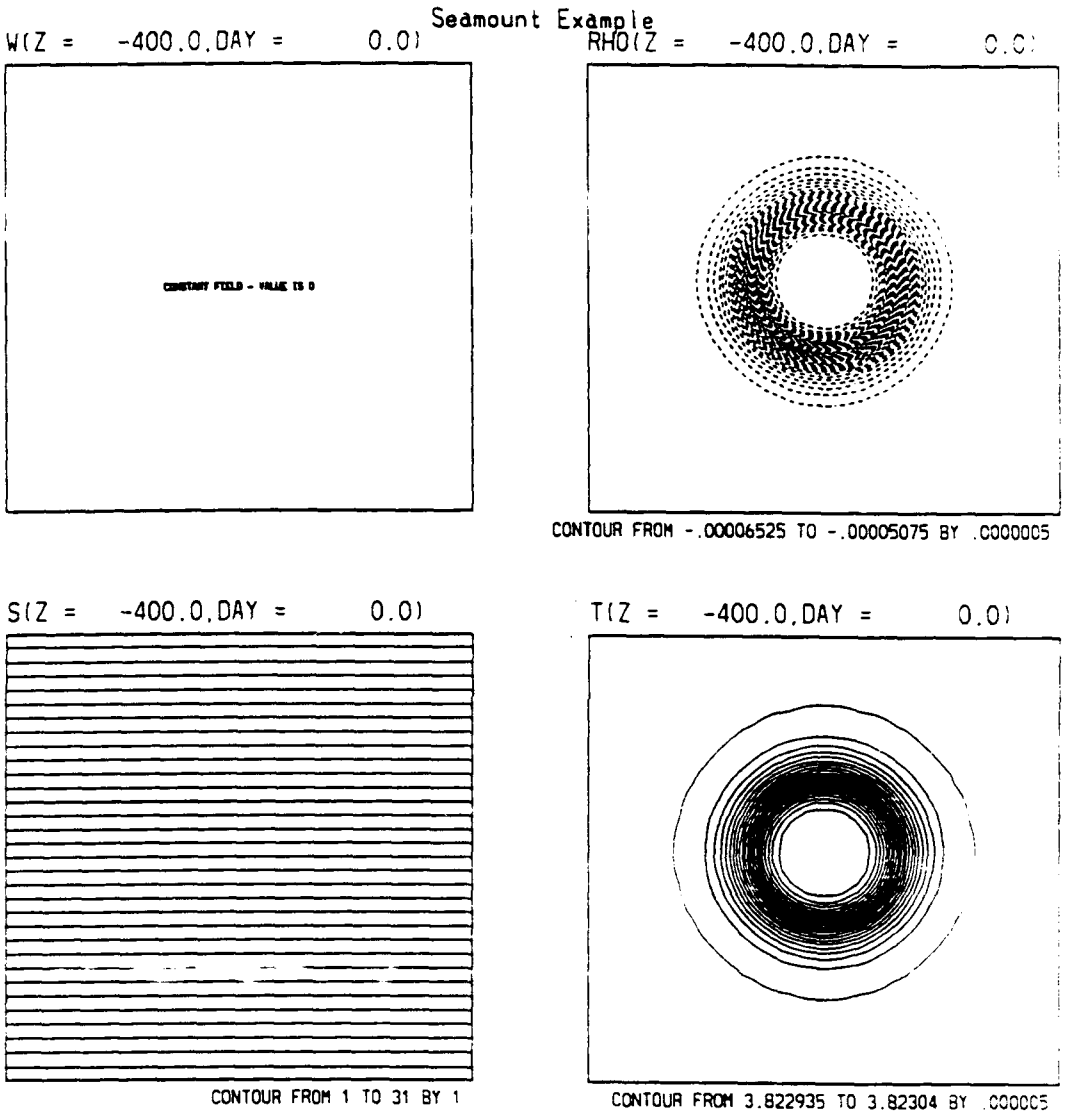


Figure 5.7: Slabs of the w, ρ, S and T fields at day 0

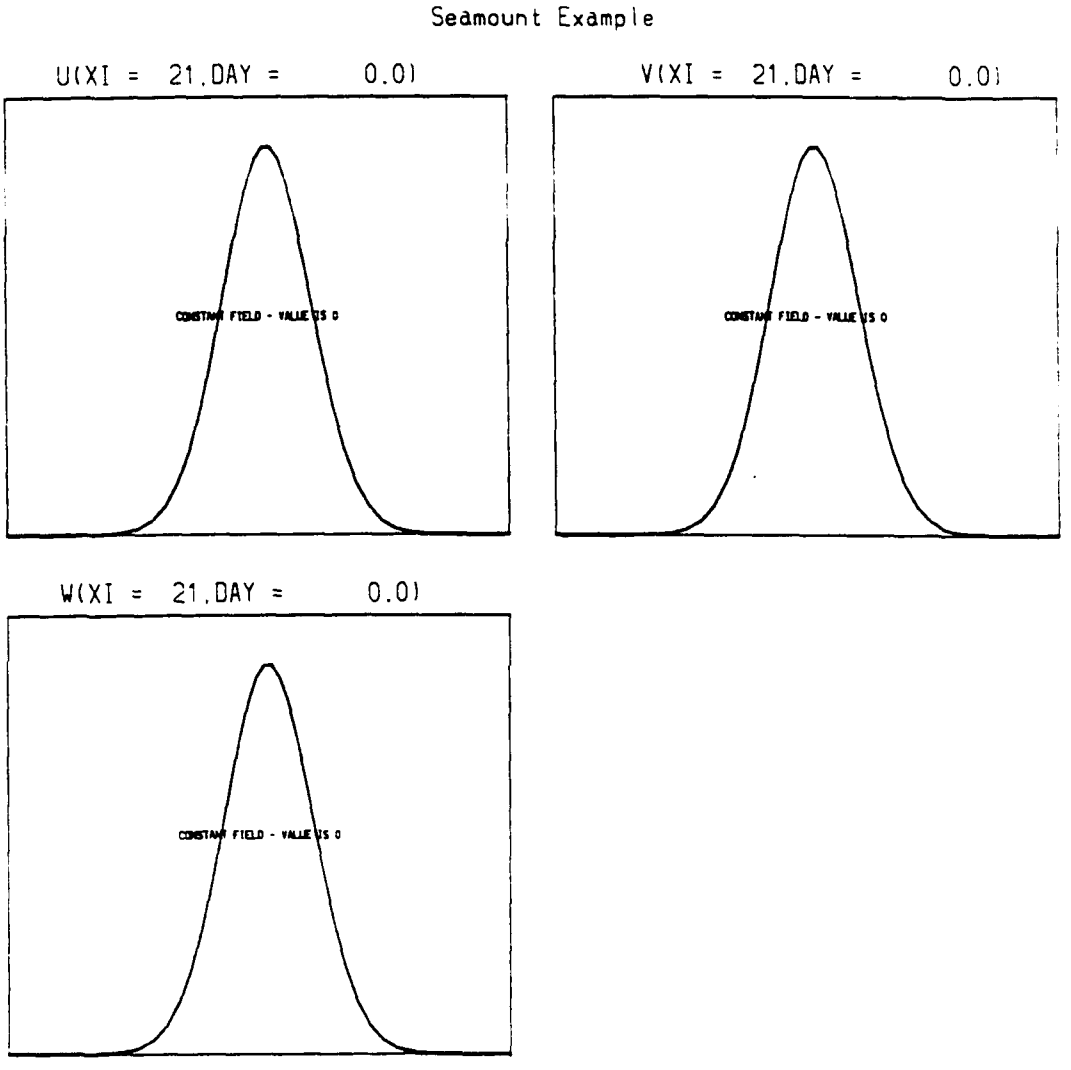


Figure 5.8: Constant ξ slices of the u, v and w fields at day 0

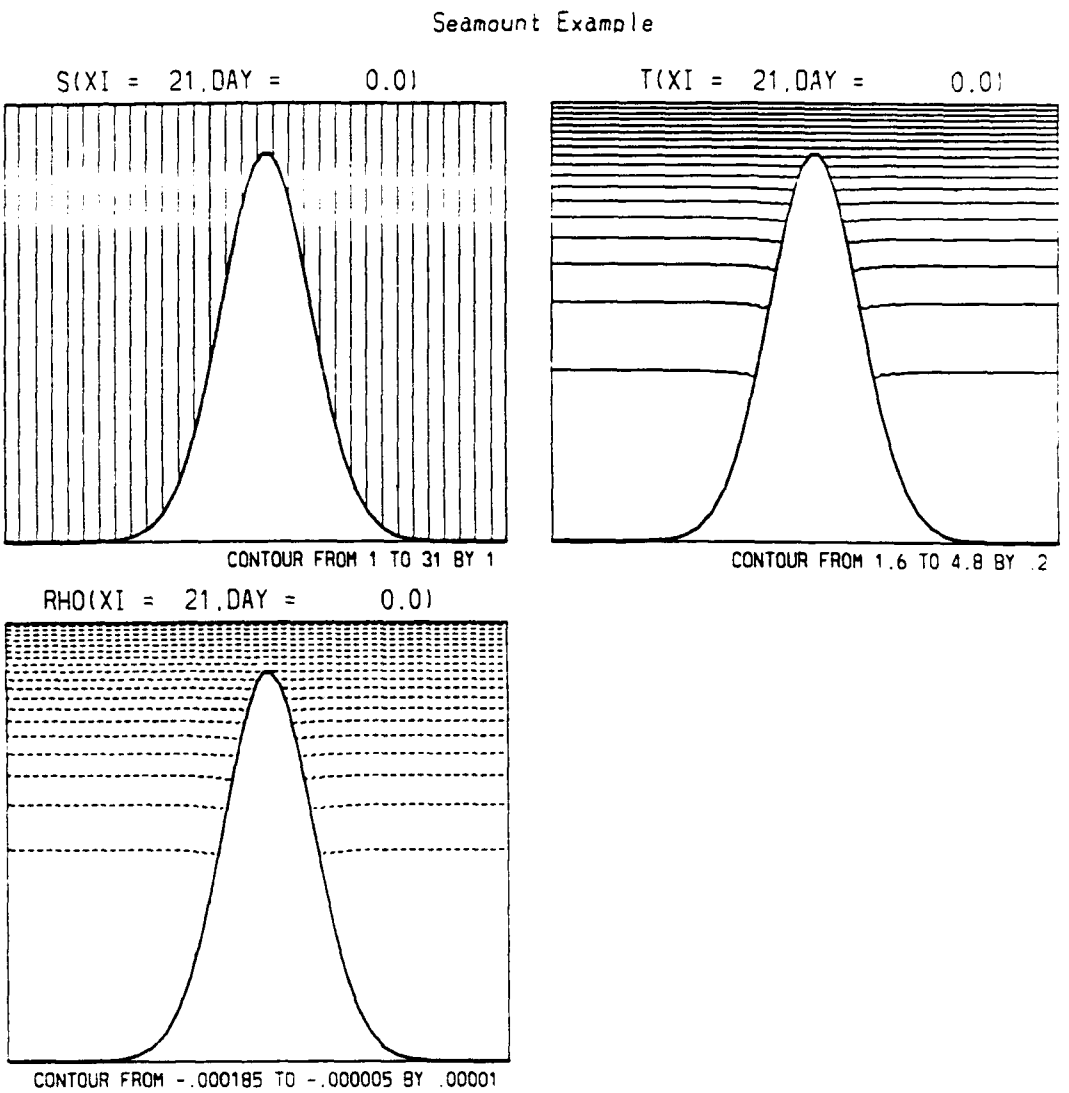


Figure 5.9: Constant ξ slices of the S, T and ρ fields at day 0

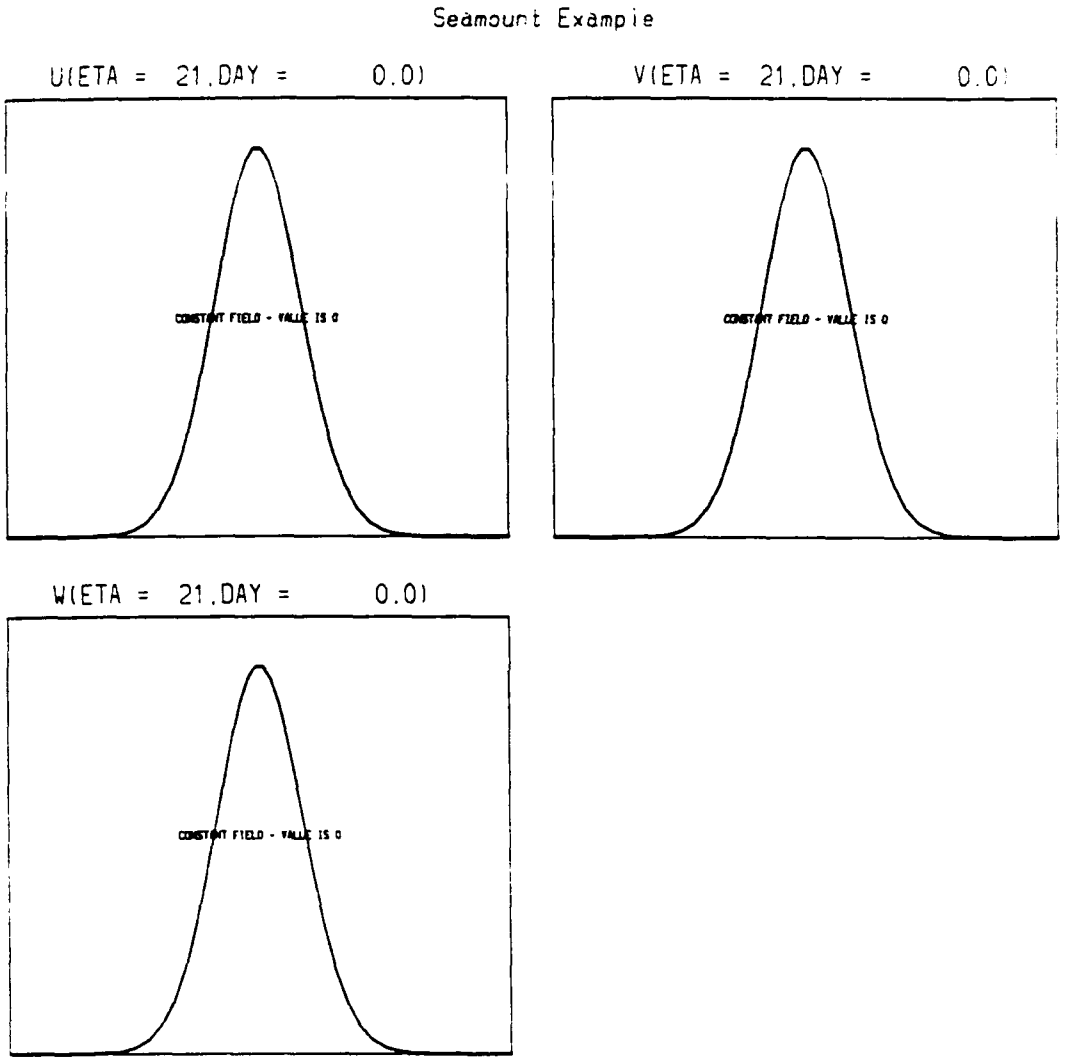


Figure 5.10: Constant η slices of the u, v and w fields at day 0

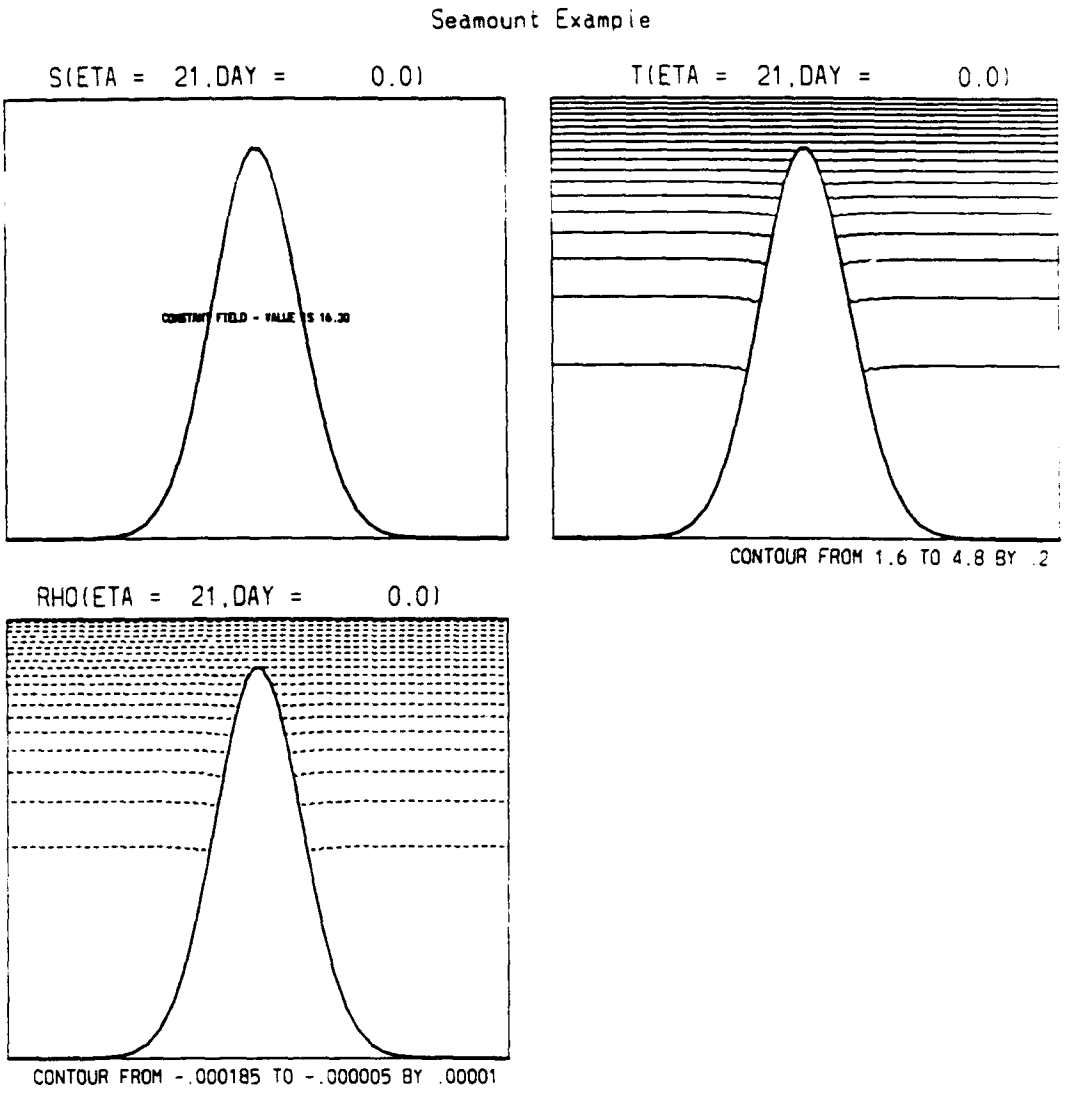


Figure 5.11: Constant η slices of the S, T and ρ fields at day 0

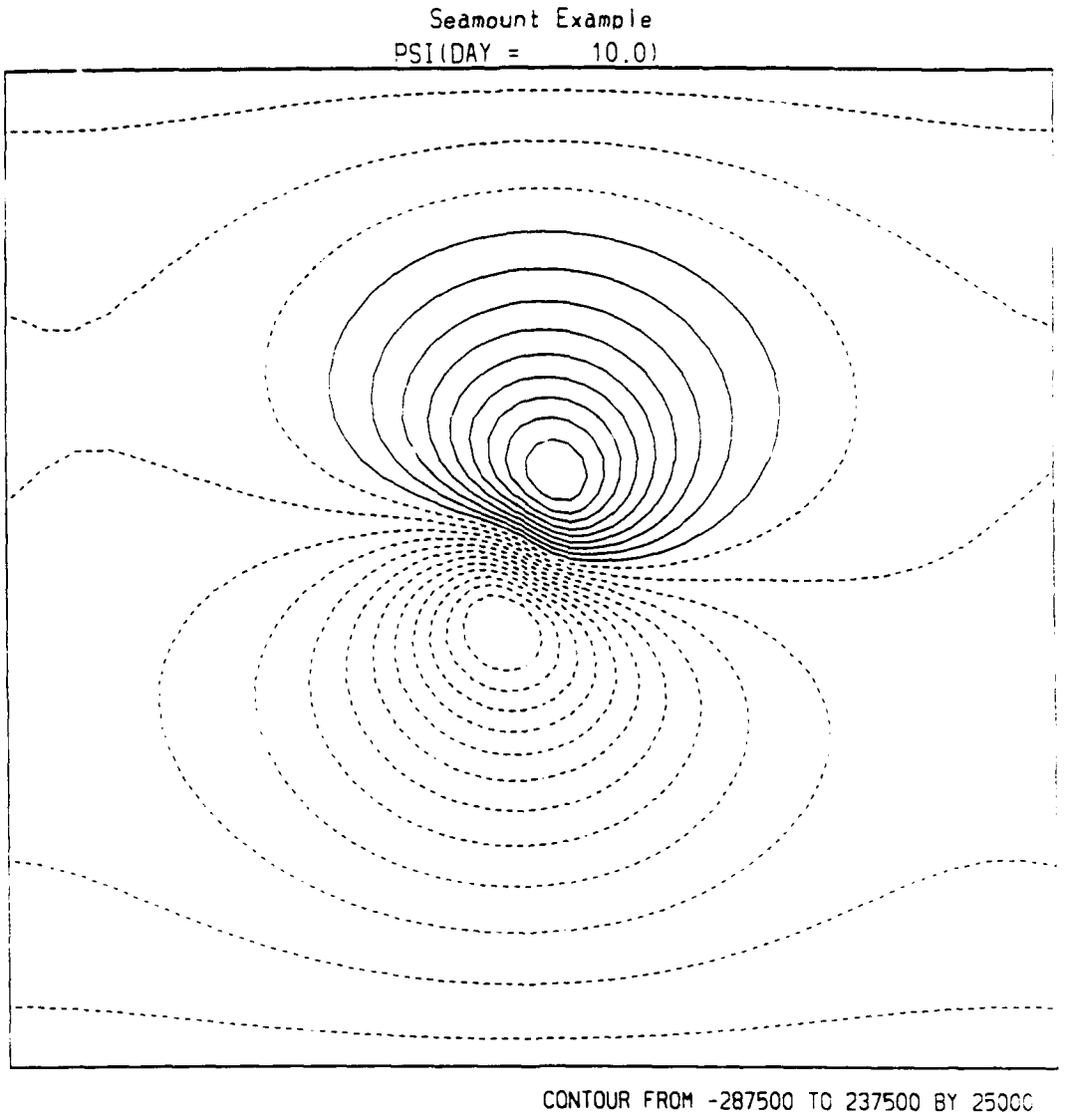


Figure 5.12: ψ field at day 10

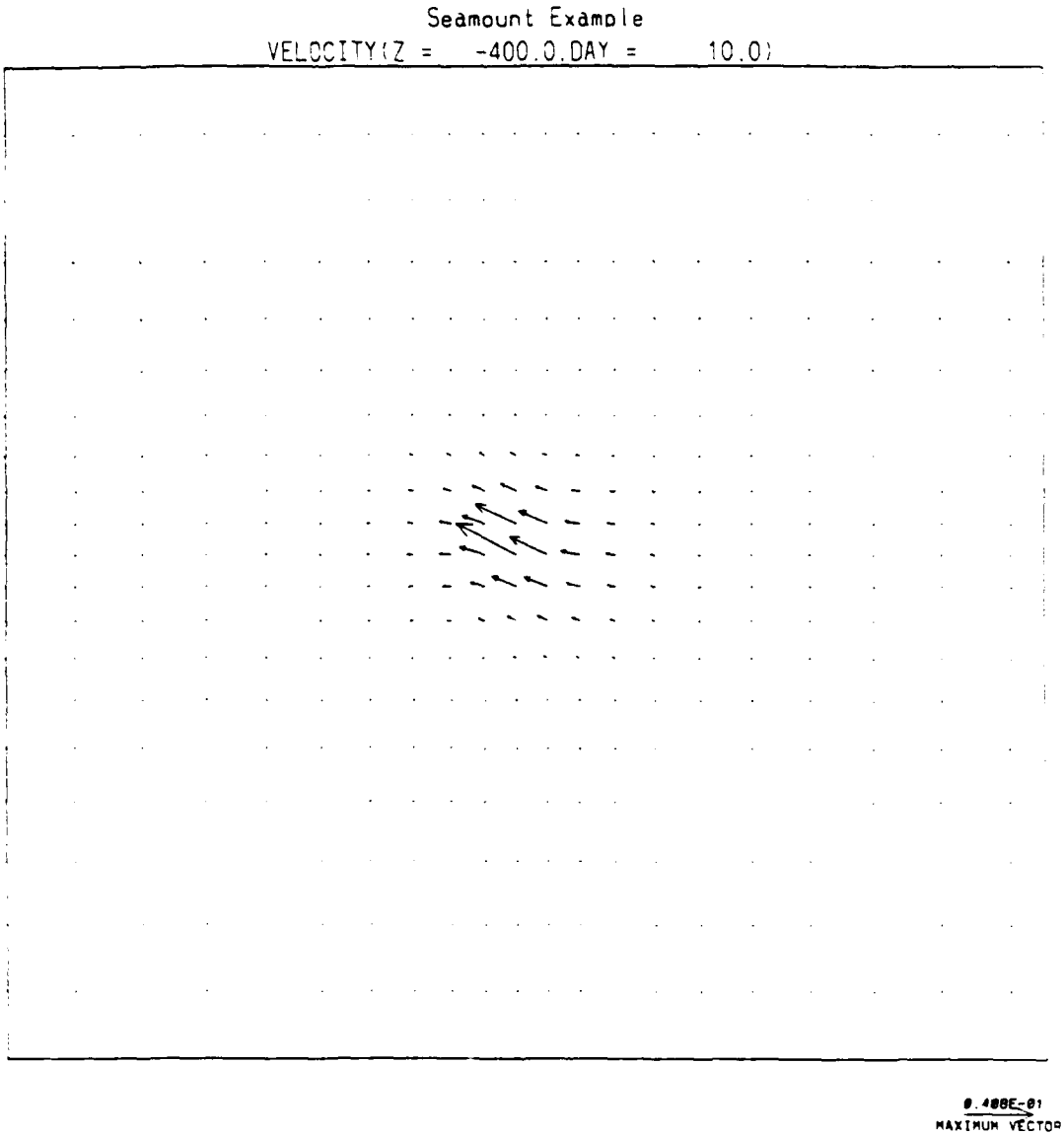


Figure 5.13: $\vec{v}$ field at day 10

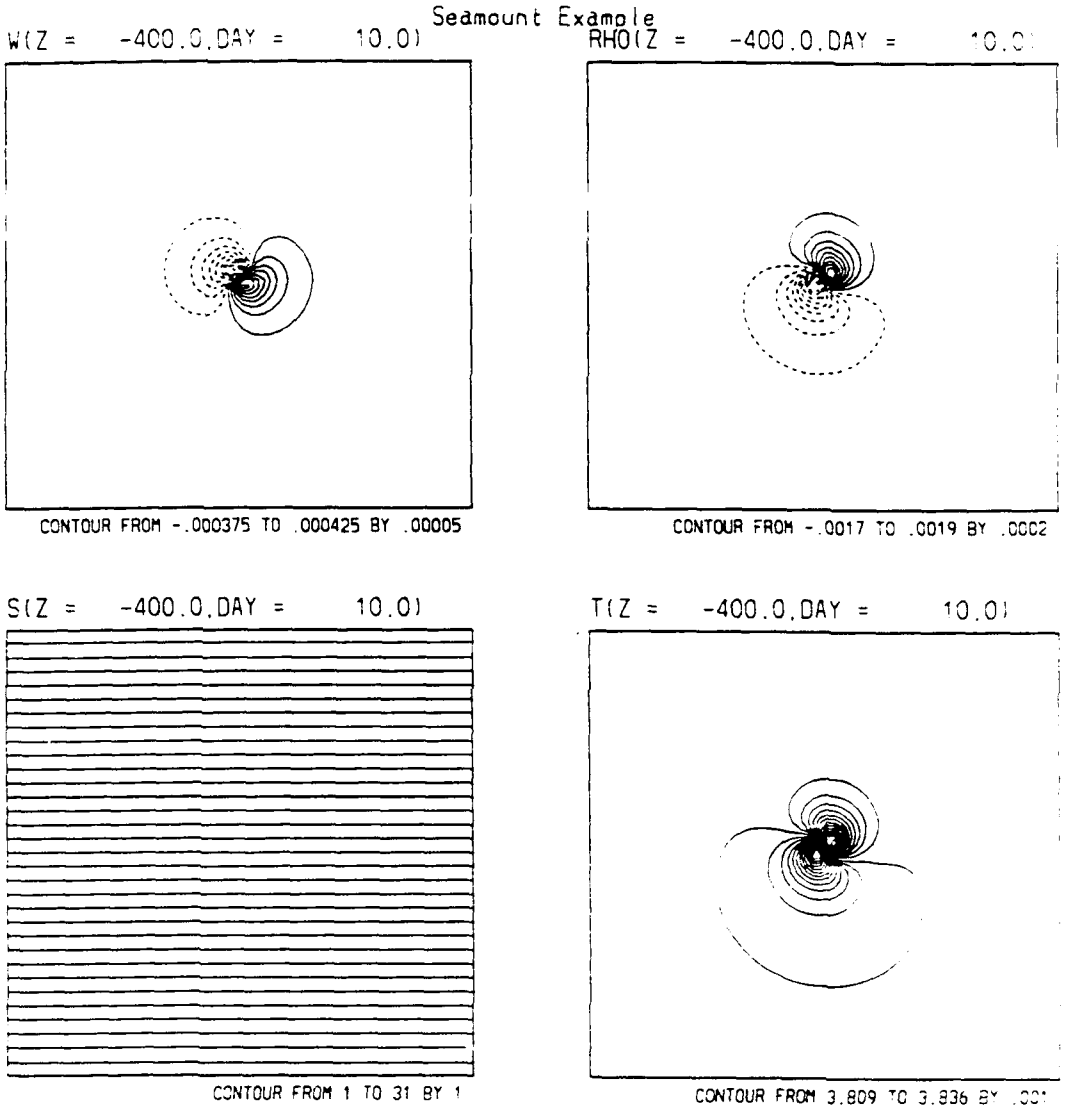


Figure 5.14: Slabs of the w, ρ, S and T fields at day 10

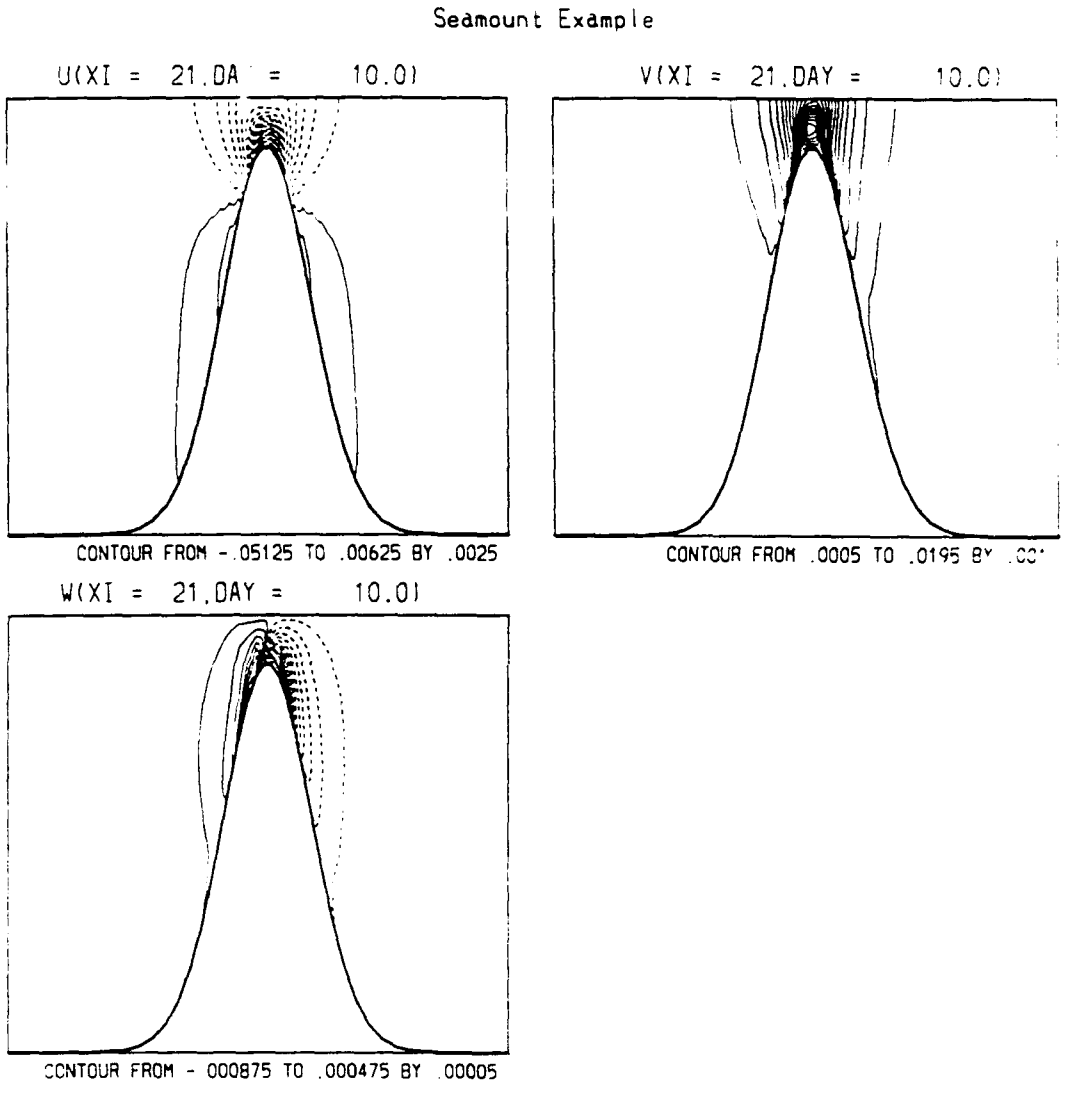


Figure 5.15: Constant ξ slices of the u, v and w fields at day 10

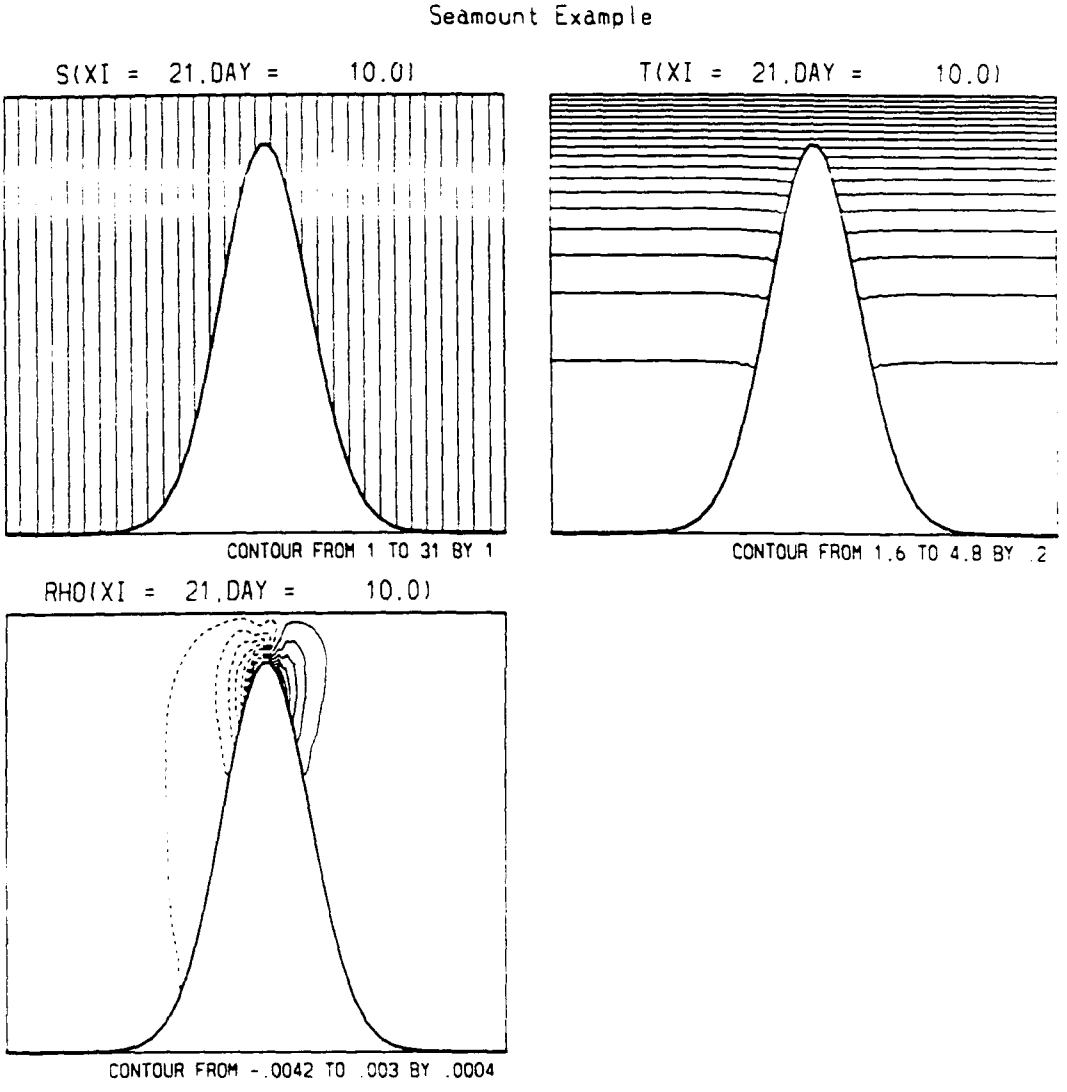


Figure 5.16: Constant ξ slices of the S, T and ρ fields at day 10

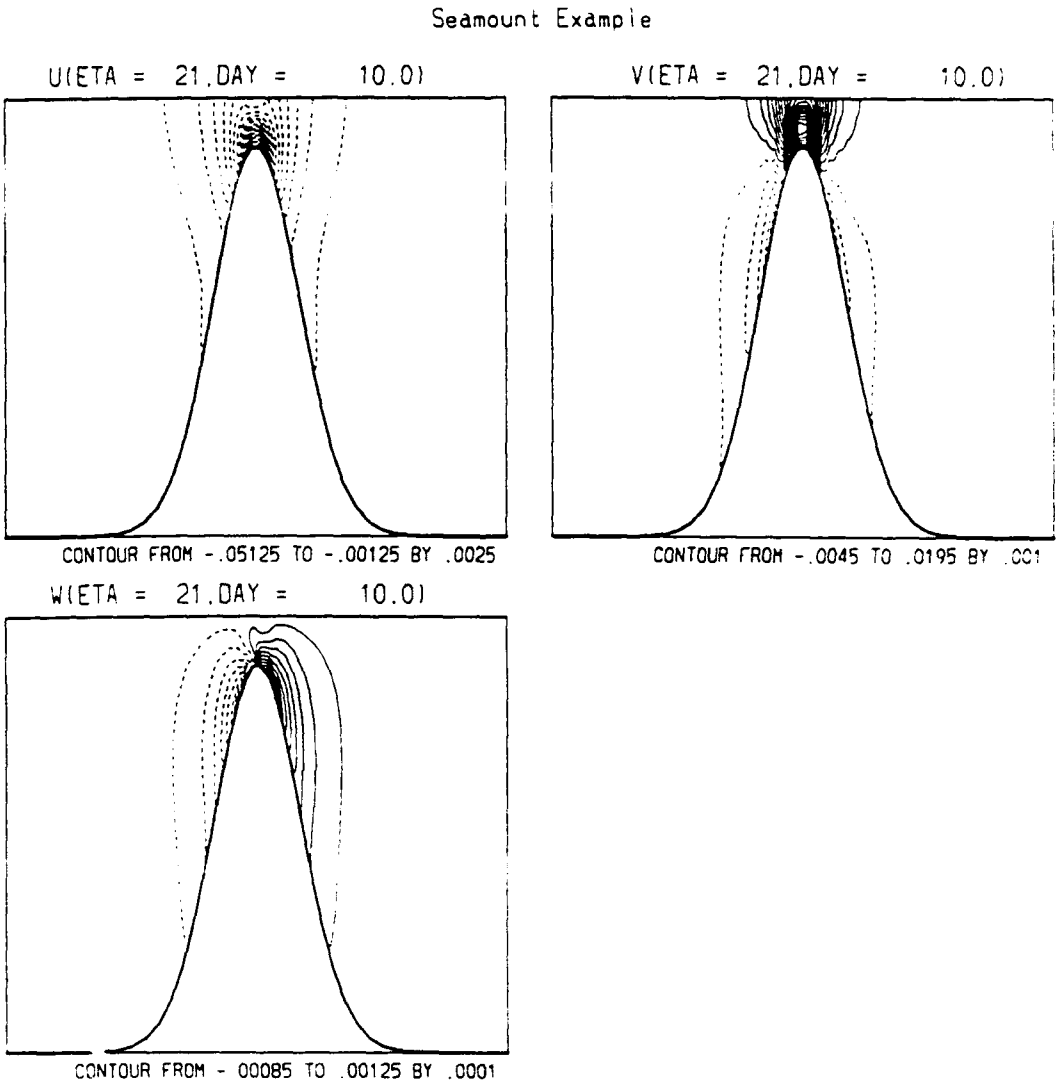


Figure 5.17: Constant η slices of the u, v and w fields at day 10

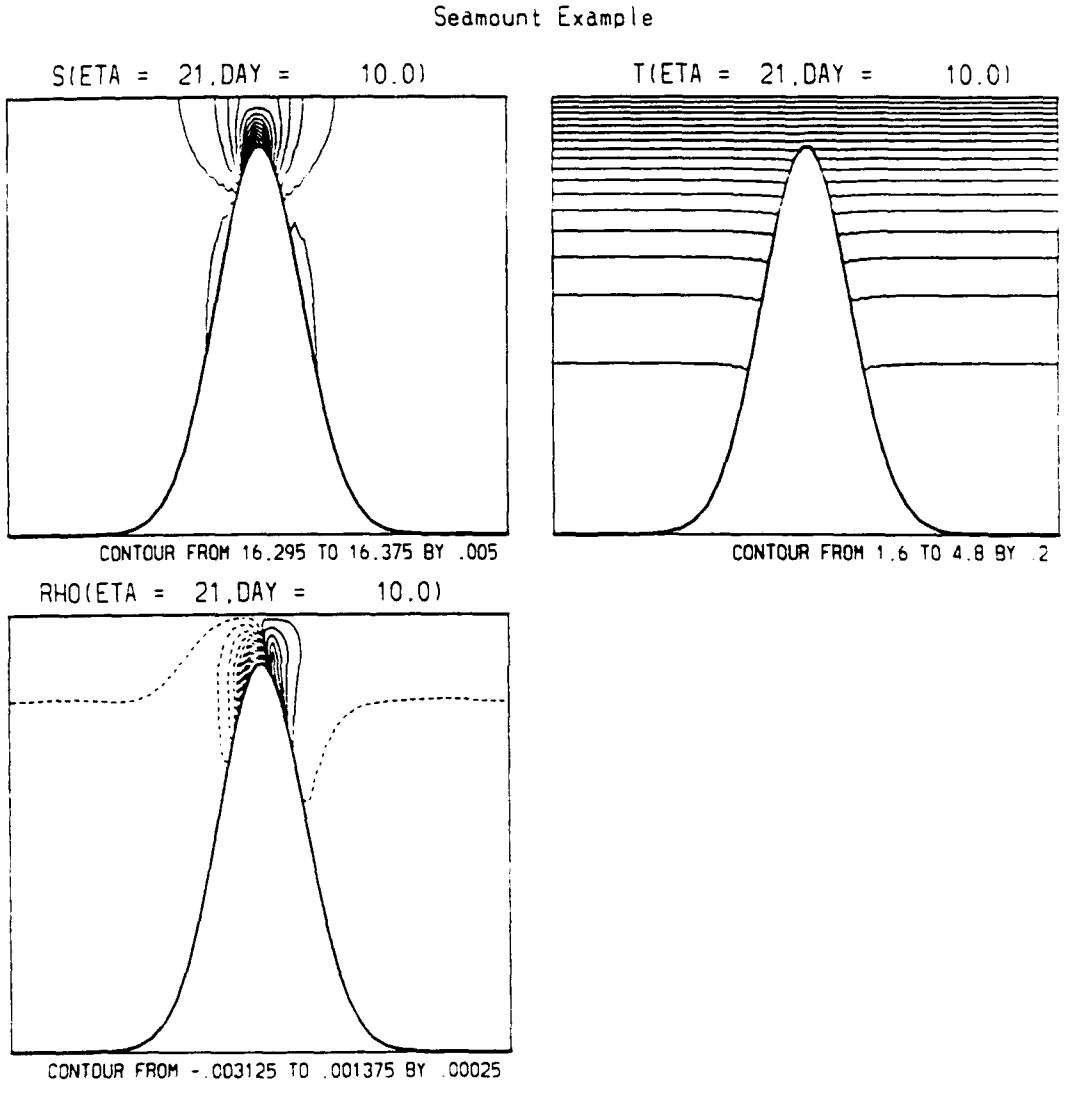


Figure 5.18: Constant η slices of the S, T and ρ fields at day 10

Chapter 6

Lagrangian Floats

6.1 General description

SPEM has optional Lagrangian floats for producing Lagrangian statistics and for simulating drifters. The float information is stored in an array and the floats are advected at each timestep by the model velocity fields. The float positions and values of T, S, ρ, and $\vec{u}$ at these positions are written out to a float history file at user-specified intervals. Plots and float statistics can be created from this history file.

The particles are advected using a fourth order Runge-Kutta scheme. For this the fluid velocity at the particle position is needed. The particles can be located between the grid points so a horizontal interpolation is done using the neighboring grid points. There is a choice of a bilinear interpolation using the four nearest grid points or a bicubic interpolation using the four by four square of grid points surrounding the particle. The algorithm for the bicubic interpolation is

$$A(x_f, y_f) = \sum_{i=1,4} \sum_{j=1,4} A(x_i, y_j) \frac{\prod_{k=1,4}^{k \neq i} (x_f - x_k) \prod_{l=1,4}^{l \neq j} (y_f - y_l)}{\prod_{k=1,4}^{k \neq i} (x_i - x_k) \prod_{l=1,4}^{l \neq j} (y_j - y_l)} \tag{6.1}$$

where (x_f, y_f) is the position of the particle and (x_i, y_j) are the positions of the 16 neighboring grid points. The interpolation is done in the horizontal plane at the depth, in sigma space, of the particle. The velocities at that depth are calculated using the polynomial modes of the model.

At each timestep, each particle is checked to see if it is inside the model domain. If a particle leaves, it is reinitialized at a fixed location which is specified by the user (at **xnew**, **ynew**, and **znew**) and a message is printed to the standard output (unit 6). If you find that particles are often leaving the box other than through open boundaries, then your timestep may be too long.

6.1.1 bcw

In the model the vertical velocity array **w** ends one half grid point to the inside of the edge of the model domain. To allow the same interpolation scheme in all of the velocity fields it was decided to add exterior points to the **w** array. The exterior values are determined in the subroutine **bcw** by a linear extrapolation from the interior values of the array. These exterior values are used only in the particle tracking subroutines.

6.2 To use floats

The changes which need to be made to the code to use floats are as follows:

6.2.1 nflt

Decide how many floats you wish to follow and set the parameter **nflt** accordingly.

6.2.2 blockdata

In the blockdata, set the variable **fltflag** to **true**. Make sure the variables **flt4flag**, **fltrrec**, and **kptime** are given appropriate values.

6.2.3 fltinit

Much like **init**, **fltinit** will either initialize the floats from a history file or it will provide initial positions for all the floats. These float positions are in the mapped model coordinate system, i.e. ξ, η, σ. The floats must lie in the range $1 \leq \xi \leq \mathbf{L}$, $1 \leq \eta \leq \mathbf{M}$, and $-1 \leq \sigma \leq 1$.

Some aspects of the float plotting program **fltplt** assume that the floats were released in logical groups, for instance, in five groups deployed at five z levels in the same pattern. However, it is not necessary to group the floats in this way.

6.2.4 fltstp

In the subroutine **fltstp**, there are the local variables **xnew**, **ynew**, and **znew**. They specify the position at which a float will be reinitialized if it leaves the model domain. Once again, these values are in the model coordinate system (ξ, η, σ). A more sophisticated scheme could be used in an open domain where you expect the floats to have a larger chance of leaving.

6.2.5 rhoflt

The subroutine **rhoflt** computes the density at the floats from the temperature and salinity at the floats. Check to make sure it is consistent with **rhocal**.

6.3 Plotting the float tracks with fltplt

At any given time, SPEM knows only the current float positions. Therefore, a separate program, called **fltplt**, was written to plot float tracks using the information in the float history file. It makes two-dimensional plots in x, y coordinates and draws the float tracks in color where color represents one of the variables z, w, T, S or ρ.

There is an option of plotting a density slab as a background to the float tracks. This option requires access to the appropriate SPEM history file and also the same equation of state which was used by SPEM.

6.3.1 Changes to fltplt

Like SPEM, **fltplt** has parameters and variables which must be changed for each application.

Parameters

nflt Number of floats.

ntm Number of float records (or larger). **fltplt** reads in all the float information which was written out by SPEM and some arrays have to be dimensioned large enough. If you wrote out float information 4 times/day and ran for 40 days, you need **ntm** to be at least 160.

nlayer One possible float release strategy is to put out an array of floats at each of several depths. All the floats at one depth would then logically belong together and are put in the same plot in the 'default' plots. The number of these float groupings is given by the parameter **nlayer**. **nlayer** should be set to 1 if a different release strategy is used.

Main program

iin, iflt, igrid Set the I/O unit numbers.

gridfile For plots without a ρ slab, the program only needs to read in the grid file (unit 3) and the float file (unit 12). If you want ρ slabs you should have a history file available. This may have a header (with **ident**, climatology, etc). If **gridfile** is **true** then **fltplt** will try to read the grid file, otherwise it will read in the header.

perchan, eqstflag Same as in SPEM. **eqstflag** is only needed when plotting ρ slabs.

fpday Number of float records per day. Calculate it from the model timestep **dt** and the frequency of saving float records **kptime**. That is,

$$\mathbf{fpday} = \frac{86400}{\mathbf{dt} \times \mathbf{kptime}}.$$

Character quality The quality of the characters drawn by **plotchar** is set by a call to **pcseti**. The best is 0, the cheapest is 2.

lwidth Sets the width of the outline drawn by pltperim and the width of the float tracks. Makes no difference on some hardware.

rhocal Make sure that you have the same **rhocal** as in SPEM.

Block data Initially, just make sure there are the proper number of values (**nlayer**) for each limit array. If you want to set the color range on the plots, the limits can be set in the block data. To use your limits, type 'n' in response to the question 'Should the range of colors be calculated from the chosen floats?'.

Colors Each of the five plots (z, w, T, S, ρ) has its own color bar, defined in **tzplt**, **twplt**, **ttplt**, **tsplt** and **trhoplt**. To change a color bar, set the parameter **ncol** in the appropriate subroutine to the number of colors you want, then fill the **rgb** array.

(Note that some compilers have a default limit to the number of allowed continuation lines).

The foreground and background colors have been set everywhere to black and white respectively. Change the calls to **gscr** if you would like them to be different.

6.3.2 Running fltplt

It will interactively ask you a number of questions, most of which should be obvious. Some which aren't are:

- 'Would you like the default plots?' Answering 'yes' will tell it to plot all the floats in each group together. For each of the **nlayer** groups, there will be a plot for each of z, w, T, S and ρ. There will be no density slabs and no more questions.
- 'Would you like to specify a range of float IDs?' This is so you can ask for floats 121 to 140 without typing in 121, 122, 123, ... It will assume that all floats to be plotted have consecutive ID's.
- 'Should the range of colors be calculated from the chosen floats?' If not, the limits set in the blockdata will be used. The limits will be that of the group to which the first float to be plotted belongs.
- 'Would you like to plot a density surface?' If so, you should know the structure of your model field history file. For example, if you want the density field from day 60, you have to know that day 60 is in the seventh restart record.

  The depth is given as a z value which is a negative number.

  For the contour interval, you can say .3 to get a contour interval of .3, 0. to accept the **conpack** default of about 16 contours, or a negative integer **ncl** to get about $-$**ncl** contours.

6.4 Seamount example

To demonstrate the use of floats, the process of putting them in the seamount resonance version from section 5.2 will be described. However, this application will not lead to the most dramatic float tracks since the flow speeds are so small.

6.4.1 In SPEM

nflt Let's put a 5 by 5 array of floats over the seamount at two depths, using 50 floats.

blockdata Set **fltflag** to **true** and **flt4flag** to **false** (for economy). **fltrrec** is 0 initially. To produce 8 float records per day, set **kptime** to 16.

fltinit The center of the seamount is at $(\xi, \eta) = (21.5, 21.5)$, which will also be the center of the float array. The float spacing will be $\Delta\xi = \Delta\eta = 2$. The first group of 25 floats will be at a depth of -100 meters while the second group will be at -499.9 meters.

fltstp The floats should not be leaving the domain so the restart location (**xnew**,**ynew**,**znew**) is not critically important. Floats will get restarted in the corner at $(\xi, \eta, \sigma) = (2, 2, .9)$.

rhoflt The equation of state in **rhoflt** is consistent with that in **rhocal**.

6.4.2 In fltplt

Parameters Set the parameters **nflt**, **ntm** and **nlayer** to 50, 81 and 2 respectively.

Main program We want to plot density slices with the float tracks so we will be reading the SPEM history file. It has a header with the grid information so **gridfile** is **false**. The SPEM history file unit is specified by **iin** and is set to 1. The float history file unit is specified by **iflt** and is set to 12.

perchan and **eqstflag** are **true** and **false** respectively. **fpday** is 8.

rhocal Use the same equation of state as in SPEM for the density slab plots.

Block data Make sure there are two values for each of the limits arrays, one for each float group.

Colors The color bars each have 14 colors, 7 reds and 7 blues.

6.4.3 Example plots

There are many options when running fltplt. To get an overview of what all the floats do, it is convenient to ask for the default plots, the first frame of which is shown in figure 6.1. It is then possible to focus in on a few floats and to overlay a density slab plot to get an idea of how the float motion relates to the physical fields, as shown in figure 6.2.

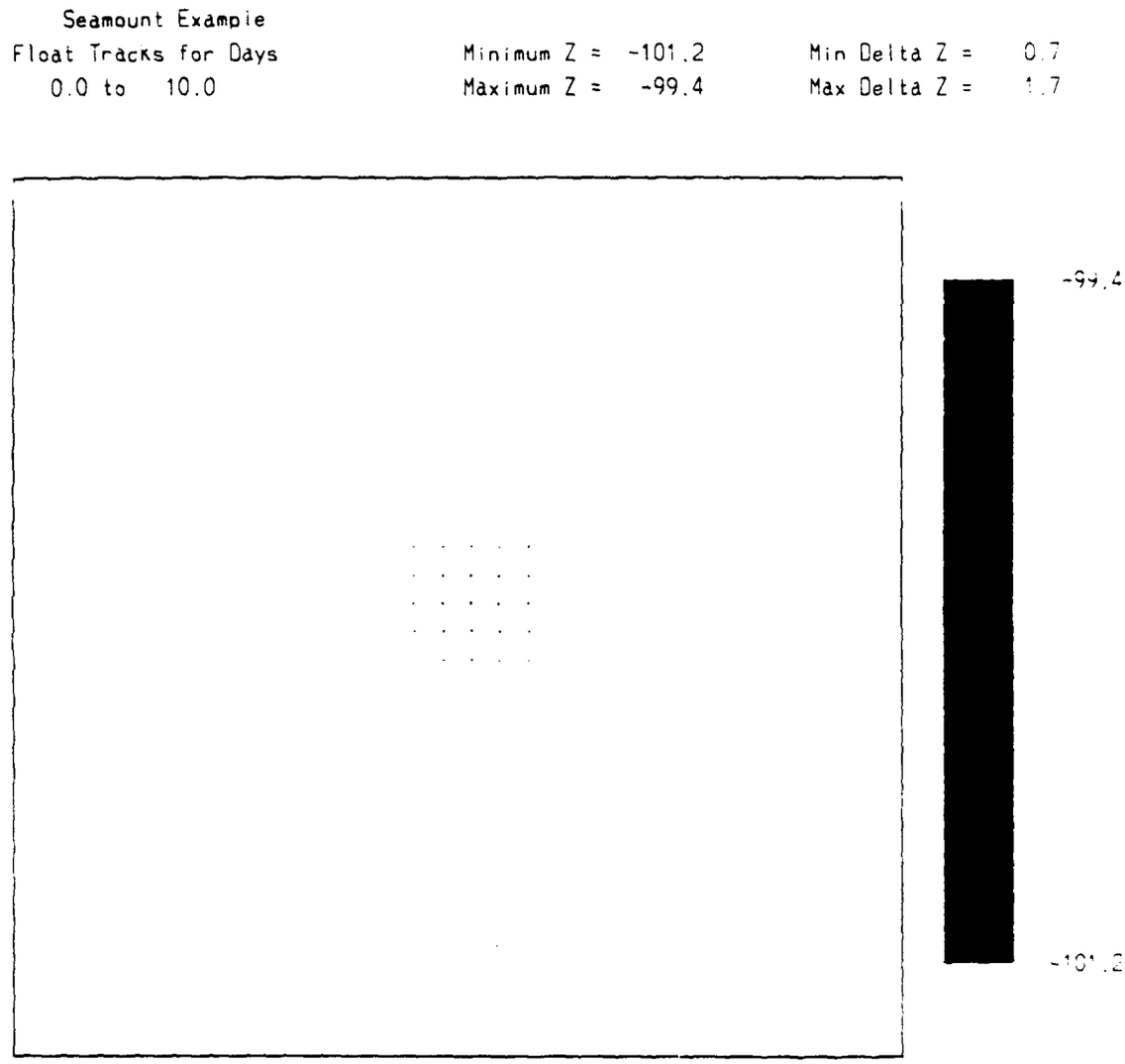


Figure 6.1: First frame of the default plots

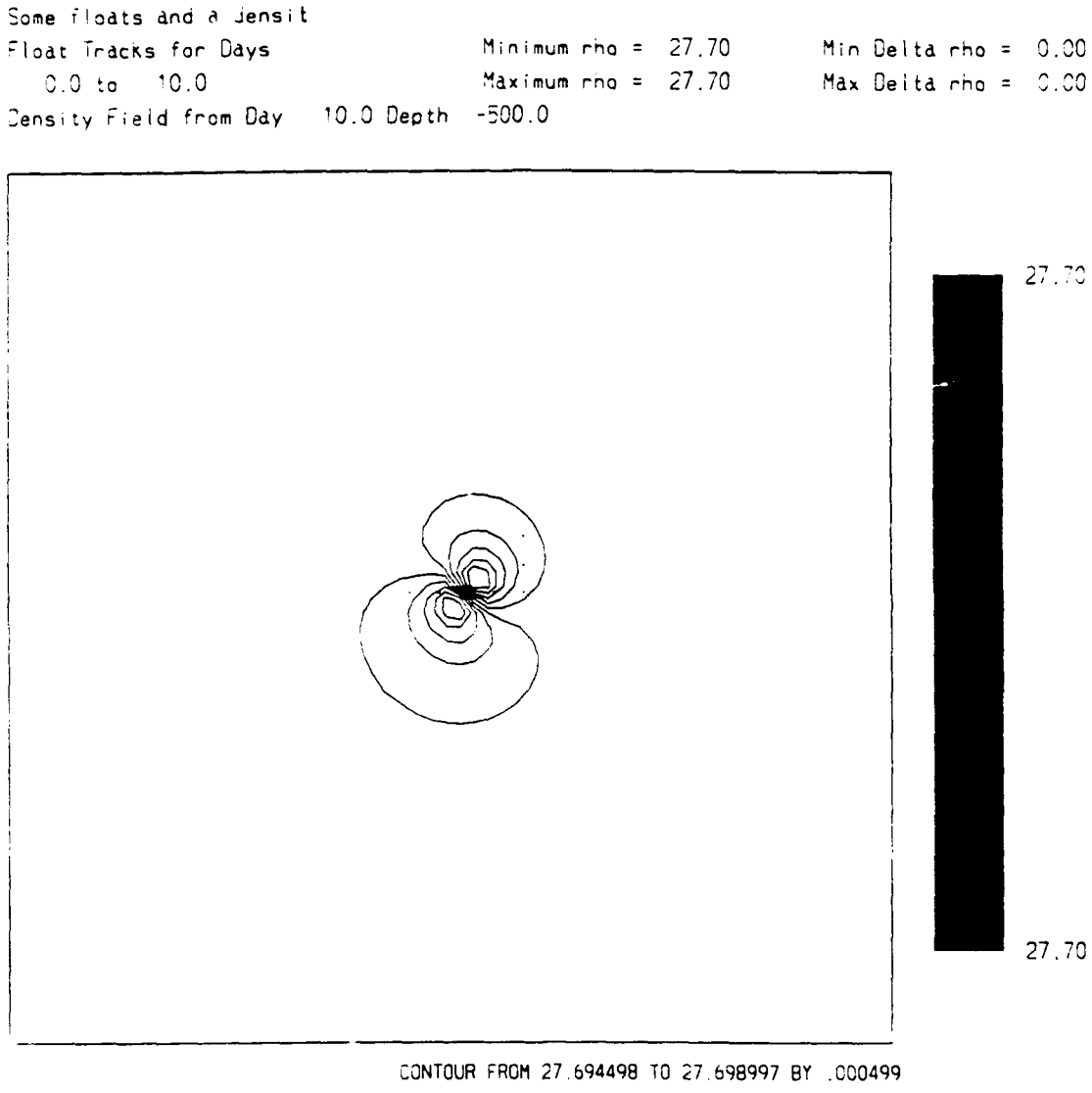


Figure 6.2: Several floats and a density surface

Appendix A

Model Time-step

Numerical time-stepping uses a discrete approximation to

$$\frac{\partial u(t)}{\partial t} = \mathcal{F}(t) \tag{A.1}$$

where $\mathcal{F}(t)$ represents all the right-hand side terms. The simplest approximation is the Euler time step:

$$\frac{u(t+\Delta t) - u(t)}{\Delta t} = \mathcal{F}(t) \tag{A.2}$$

where you predict the next u value based only on the current fields. This method is accurate to first order in Δt and is stable for a sufficiently short time-step, dictated by a combination of the advective speeds, friction coefficients, and the horizontal and vertical grid spacings.

The leapfrog time-step is accurate to $O(\Delta t^2)$:

$$\frac{u(t+\Delta t) - u(t-\Delta t)}{2\Delta t} = \mathcal{F}(t). \tag{A.3}$$

This time-step is more accurate, but the even and odd time steps tend to diverge in a computational mode. This computational mode can be damped by taking an occasional correction step. The SPEM has a choice of correction steps, both of which use a leapfrog step to obtain an initial guess of $u(t+\Delta t)$. We will call the right-hand side terms calculated from this initial guess $\mathcal{F}^*(t+\Delta t)$.

The first choice of correction steps is a trapezoidal step:

$$\frac{u(t+\Delta t) - u(t)}{\Delta t} = \frac{1}{2}\left[\mathcal{F}(t) + \mathcal{F}^*(t+\Delta t)\right] \tag{A.4}$$

and the second choice is a third-order Adams-Bashforth:

$$\frac{u(t+\Delta t) - u(t)}{\Delta t} = \mathcal{F}(t) - \frac{1}{4}\mathcal{F}(t-\Delta t) + \frac{1}{4}\mathcal{F}^*(t+\Delta t). \tag{A.5}$$

The SPEM uses a leapfrog time step with a correction every **ncorr** time steps. The type of correction step is determined by the value of **stpflag** (**true** for a third-order Adams-Bashforth).

The model carries two time levels for many of the physical fields, differentiated by the last index in the arrays. The initial fields are only read in for one time level; an Euler time step is used for the first step to get things going. A correction step is made if you are using trapezoidal corrections (you would need to obtain $\mathcal{F}(-\Delta t)$ to make a third order correction).

Every **nrst** time steps a restart record is written. If you were to start the model from one of these restart records you would start with an Euler step. Therefore, to give repeatable results, the model treats the first step after each record is written as if it were the first step.

Appendix B

Chebyshev Polynomial Basis Functions

As described in section 3.1, the $P_k(\sigma)$ expansion functions used in the vertical spectral method can be chosen somewhat arbitrarily. Here we review the form of the matrix operators when the $P_k(\sigma)$ are a modified form of the Chebyshev polynomials of the first kind ($T_k(\sigma)$: see, e.g., [1] and [9]). The $T_k(\sigma)$ are defined as

$$T_k(\sigma) = \cos[k \cos^{-1}(\sigma)], \qquad -1 \leq \sigma \leq 1$$

where $-\pi \leq cos^{-1}(\sigma) \leq 0$. We then set

$$P_k(\sigma) = \begin{cases} T_0(\sigma) & k = 0 \\ T_k(\sigma) & k \geq 1, \quad k \text{ odd} \\ T_k(\sigma) + \frac{1}{k^2-1} & k \geq 2, \quad k \text{ even.} \end{cases} \tag{B.1}$$

which then conform to the required integral property that only $P_o(\sigma)$ have a non-zero vertical integral. The collocation points σ_n are located at the extrema of the highest order polynomial $P_N(\sigma)$; hence,

$$\sigma_n = \cos[\pi(n - N)/N], \qquad 0 \leq n \leq N. \tag{B.2}$$

The first eight modified Chebyshev polynomials and the collocation levels for $N = 8$ are shown in figure B.1.

The matrix F (see equation (3.3)) may now be evaluated form (B.1) and (B.2). Substituting (B.1) into (3.6) gives the elements of matrix R:

$$R_{nk} = \frac{\partial P_k(\sigma)}{\partial \sigma} = \frac{k \sin(k\theta_n)}{\sin \theta_n} \tag{B.3}$$

where $\theta_n = \cos^{-1}(\sigma_n)$. Similarly, (B.1) and (3.8) give the elements of matrix S:

$$S_{nk} = \int_{\sigma_n}^{1} P_k(\sigma) d\sigma = \begin{cases} (1 - \sigma_n) & k = 0 \\ \frac{1}{2}(1 - \sigma_n^2) & k = 1 \\ \left[\frac{\cos(k+1)\sigma}{2(k+1)} - \frac{\cos(k-1)\sigma}{2(k-1)}\right]_{\cos^{-1}(\sigma_n)}^{0} & k > 1, \text{odd} \\ \left[\frac{\cos(k+1)\sigma}{2(k+1)} - \frac{\cos(k-1)\sigma}{2(k-1)}\right]_{\cos^{-1}(\sigma_n)}^{0} & \\ \quad + (1 - \sigma_n)\frac{1}{k^2-1} & k > 1, \text{even.} \end{cases} \tag{B.4}$$

Finally, the differentiating and integrating operators C_{DZ} and C_{INT} are computed by first inverting F and calculating the products $C_{DZ} = DF^{-1}$ and $C_{INT} = SF^{-1}$.

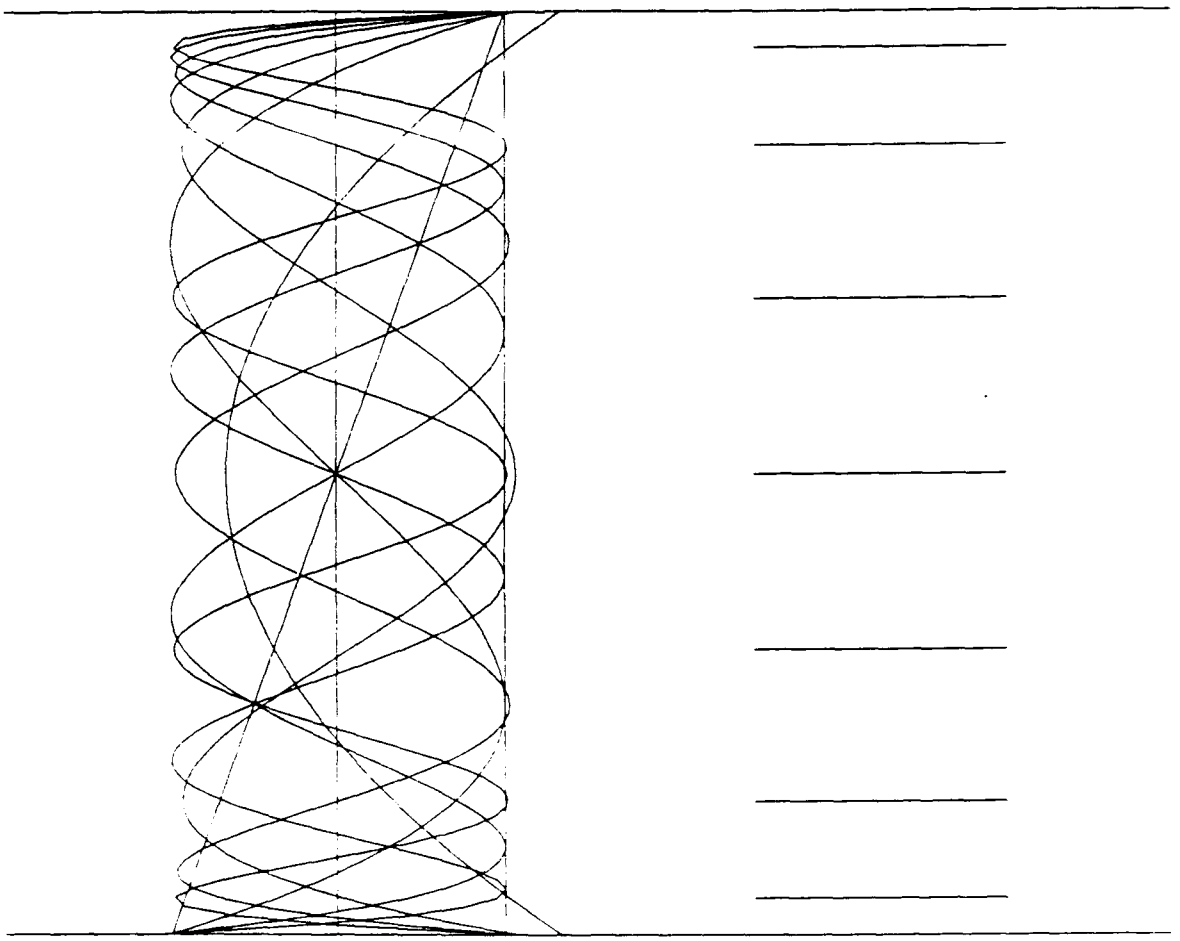


Figure B.1: Modified Chebyshev polynomials and collocation levels for $N = 8$

Appendix C

Topographic Steepness

All sigma coordinate models are known to have some trouble in the presence of steep topography, especially if the grid spacing is not fine enough. Unfortunately, we do not know in general the relationship between steep topography, stratification, horizontal and vertical resolution and whether the model will run stably. However, two plots have been added to **ploth** to guide the user in determining which areas of the domain are likely to lead to topographic problems.

The first of these plots is simply the bottom slope, or $|\nabla h|$. The maximum for the whole domain is written in the corner of the plot. We have successfully run the SPEM with bottom slopes of up to ~ 0.10, while continental shelves can have slopes in excess of 0.25.

If you find that the model is ill-behaved and you suspect a problem with steep topography, there are two ways to improve on the situation. One is to filter the topography until it is no longer 'too steep'. The other is to increase the horizontal grid spacing until you resolve the slope 'well enough'. A measure of how well the slope has to be resolved is gotten from the "$\mathcal{R}$-value" (the ratio of horizontal to vertical resolution):

$$\mathcal{R} = \left| \frac{\Delta h}{\Delta(h\sigma)} \right|$$

This is a three-dimensional quantity, but we only plot it as a function of x and y since we know that when using the Chebyshev polynomials, the maximum is between the two collocation levels closest to the bottom.

$\mathcal{R}$ is related to the "hydrostatic consistency" requirement known from discrete level models, where $\mathcal{R}$ has to be less than 1, but SPEM has run with values up to 5. Once again, we cannot tell exactly what will or will not run. These tools are provided to help diagnose problems with steep topography.

Appendix D

Context Diffs for velvct

What follows is a Unix context difference file. It starts with a header specifying which files are being compared. The lines which differ are shown with a few surrounding lines for context and are marked with an exclamation point. The lines from the distributed **velvct.f** are shown first, followed by the lines from the modified version. See section 4.5 for a description of the changes.

```
diff -c velvct/velvct.f velvct.mod/velvct.f
*** velvct/velvct.f Fri Jan 26 09:24:17 1990
--- velvct.mod/velvct.f Mon Mar 19 09:52:58 1990
***************
*** 406,416 ****
  C                                     ILAB IS NON-ZERO.
  C
        SAVE
!       FX(XX,YY) = XX
!       FY(XX,YY) = YY
        DIST(XX,YY) = SQRT(XX*XX+YY*YY)
!       MXF(XX,YY,UU,VV,SFXX,SFYY,MXX,MYY) = MXX+IFIX(SFXX*UU)
!       MYF(XX,YY,UU,VV,SFXX,SFYY,MXX,MYY) = MYY+IFIX(SFYY*VV)
        SCALEX(MM,NN,INCXX,INCYY,HAA,XX1,XX2,YY1,YY2,XX3,XX4,YY3,YY4,
       1       LENN) = LENN/HAA
        SCALEY(MM,NN,INCXX,INCYY,HAA,XX1,XX2,YY1,YY2,XX3,XX4,YY3,YY4,
--- 406,416 ----
  C                                     ILAB IS NON-ZERO.
  C
        SAVE
! c     FX(XX,YY) = XX
! c     FY(XX,YY) = YY
        DIST(XX,YY) = SQRT(XX*XX+YY*YY)
! c     MXF(XX,YY,UU,VV,SFXX,SFYY,MXX,MYY) = MXX+IFIX(SFXX*UU)
! c     MYF(XX,YY,UU,VV,SFXX,SFYY,MXX,MYY) = MYY+IFIX(SFYY*VV)
        SCALEX(MM,NN,INCXX,INCYY,HAA,XX1,XX2,YY1,YY2,XX3,XX4,YY3,YY4,
       1       LENN) = LENN/HAA
        SCALEY(MM,NN,INCXX,INCYY,HAA,XX1,XX2,YY1,YY2,XX3,XX4,YY3,YY4,
***************
*** 551,556 ****
--- 551,561 ----
        ZMN = LEN*(GL/HA)
```

```
        ZMX = FLOAT(LEN)+.01
  C
+ c  turn off clipping so vectors can go out of domain
+ c
+       CALL GQCLIP(IER,ICLP,IAR)
+       CALL GSCLIP(0)
+ c
  C DRAW THE VECTORS.
  C
        DO 123 J=1,NY,INCY
***************
*** 580,587 ****
  C
  C     TURN OFF CLIPPING SO ARROW CAN BE DRAWN
  C
!       CALL GQCLIP(IER,ICLP,IAR)
!       CALL GSCLIP(0)
        CALL DRWVEC (28768,608,28768+LEN,608,LABEL,10)
  C
  C     RESTORE CLIPPING
--- 585,592 ----
  C
  C     TURN OFF CLIPPING SO ARROW CAN BE DRAWN
  C
! c     CALL GQCLIP(IER,ICLP,IAR)
! c     CALL GSCLIP(0)
        CALL DRWVEC (28768,608,28768+LEN,608,LABEL,10)
  C
  C     RESTORE CLIPPING
```

Bibliography

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions.* Number 55 in Applied Mathematics Series. National Bureau of Standards, 1972.

[2] J. Adams. Mudpack: Multigrid fortran software for the efficient solution of linear elliptic partial differential equations. *Applied Math. and Comput.*, 34:113–146, November 1989.

[3] J. Adams. *MUDPACK: Multigrid Software for Linear Elliptic Partial Differential Equations, Version 2.* National Center for Atmospheric Research, Boulder, Colorado, February 1990. Scientific Computing Division User Doc.

[4] A. Arakawa and V. R. Lamb. *Methods of Computational Physics*, volume 17, pages 174–265. Academic Press, 1977.

[5] F. Clare and D. Kennison. *NCAR Graphics Guide to New Utilities, Version 3.00.* National Center for Atmospheric Research, Boulder, Colorado, October 1989.

[6] F. Clare, D. Kennison, and R. Lackman. *NCAR Graphics User's Guide, Version 2.0.* National Center for Atmospheric Research, Boulder, Colorado, August 1987.

[7] N. G. Freeman, A. M. Hale, and M. B. Danard. A modified sigma equations' approach to the numerical modeling of great lake hydrodynamics. *Journal of Geophysical Research*, 77(6):1050–1060, 1972.

[8] D. Haidvogel, J. Wilkin, and R. Young. A semi-spectral primitive equation ocean circulation model using vertical sigma and orthogonal curvilinear horizontal coordinates. *Journal of Computational Physics*, 1990. (in press).

[9] D. B. Haidvogel and T. Zang. The accurate solution of poisson's equation by expansion in chebyshev polynomials. *Journal of Computational Physics*, 30(2):167–180, 1979.

[10] K. Hedstrom and J. Wilkin. *Unfinished Manuscript on Orthogonal Grid Generation.* Institute for Naval Oceanography, Stennis Space Center, Mississippi, 1990.

[11] N. A. Phillips. A coordinate system having some special advantages for numerical forecasting. *Journal of Meteorology*, 14(2):184–185, 1957.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes, The Art of Scientific Computing.* Cambridge University Press, 1986.

[13] Ralph Shapiro. Smoothing, filtering, and boundary effects. *Reviews of Geophysics and Space Physics*, 8(2):359–387, May 1970.

DISTRIBUTION LIST

1. Office of Naval Research
   Code 1242 (10 copies)
   800 North Quincy Street
   Arlington, VA 22217-5000

2. Director, Atmospheric Sciences Directorate
   NOARL West (Code 400)
   Monterey, CA 93943-5000

3. Commanding Officer
   Fleet Numerical Oceanography Office
   Monterey, CA 93943-5000

4. Commanding Officer
   Naval Oceanographic Office
   Stennis Space Center, MS 39529

5. Technical Director
   CNOC (Code OOT)
   Stennis Space Center, MS 39529

6. Commanding Officer
   NOARL (Code 100)
   Stennis Space Center, MS 39529

7. Technical Director
   NOARL (Code 110)
   Stennis Space Center, MS 39529

8. Director, Ocean Sciences Directorate
   NOARL (Code 300)
   Stennis Space Center, MS 39529

9. Head, Ocean Sensing and Prediction Division
   NOARL (Code 320)
   Stennis Space Center, MS 39529

10. Library
    NOARL (Code 125)
    Stennis Space Center, MS 39529

11. Prof. Mayer Humi
    Math Department, WPI
    100 Institute Road
    Worcester, MA 01609-2280

12. Dr. Dale Haidvogel
    The Johns Hopkins University
    Chesapeake Bay Institute
    The Rotunda, Suite 340
    711 West 40th Street
    Baltimore, MD 21211

13. Paola Rizzoli
    Dept. of Earth, Atmospheric & Planetary Science
    Massachusetts Inst. of Technology
    Cambridge, MA 02139

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. Agency Use Only *(Leave blank).*

2. Report Date. July 1990

3. Report Type and Dates Covered.

4. Title and Subtitle.
User's Manual for a Semi-Spectral Primitive Equation Regional Ocean Circulation Model Version 3.0 B

5. Funding Numbers.
Program Element No. 0602435N
Project No. RM35G92
Task No. 801
Accession No. DN250037

6. Author(s).
Katherine S. Hedstrom

7. Performing Organization Name(s) and Address(es).
Institute for Naval Oceanography
Stennis Space Center, MS 39529-5005

8. Performing Organization Report Number.
INO Technical Note FY90-2

9. Sponsoring/Monitoring Agency Name(s) and Address(es).
Office of Naval Research
800 North Quincy Street
Code 120M
Arlington, VA 22217

10. Sponsoring/Monitoring Agency Report Number.
INO Technical Note FY90-2

11. Supplementary Notes.

12a. Distribution/Availability Statement.
Approved for public release; distribution is unlimited.

12b. Distribution Code.

13. Abstract *(Maximum 200 words).*

The Semi-Spectral Primitive Equation Model (SPEM), authored by Dr. Dale Haidvogel of The Johns Hopkins University, is one approach to region and basin scale ocean modeling. This user's manual for SPEM describes the model equations as well as what the user must do to configure the model for his specific application.

14. Subject Terms.
(U) User's Manual (U) SPEM (U) Ocean Model (U) INO

15. Number of Pages. 82

16. Price Code.

17. Security Classification of Report.	18. Security Classification of This Page.	19. Security Classification of Abstract.	20. Limitation of Abstract.
Unclassified	Unclassified	Unclassified	